ÉVALUATION DES CONNAISSANCES PROCÉDURALES DANS LES RÉPONSES EN TEXTE LIBRE PAR UNE APPROCHE HYBRIDE EMPLOYANT ONTOLOGIES ET RÉSEAUX SÉMANTIQUES

THÈSE PRÉSENTÉE À LA FACULTÉ DES ÉTUDES SUPÉRIEURES ET DE LA RECHERCHE EN VUE DE L'OBTENTION DE LA MAÎTRISE ÈS SCIENCES EN INFORMATIQUE

ERIC SNOW

DÉPARTEMENT D'INFORMATIQUE
FACULTÉ DES SCIENCES
UNIVERSITÉ DE MONCTON

Janvier 2015

COMPOSITION DU JURY

Président du jury :

Éric Hervet, Ph. D.

Professeur d'informatique, Université de Moncton

Examinateur externe :

Bruno Emond, Ph. D.

Agent de recherche senior
Conseil national de recherches Canada, Ottawa

Examinateur interne :

Guillaume Durand, Ph. D.

Agent de recherche
Conseil national de recherches Canada, Moncton
Professeur associé, Université de Moncton

Directrice de thèse :

Chadia Moghrabi, Ph. D.

Professeure d'informatique, Université de Moncton

Codirecteur de thèse :

Philippe Fournier-Viger, Ph. D.

Professeur d'informatique, Université de Moncton

REMERCIEMENTS

Je tiens tout d'abord à remercier mes directeurs de thèse, les professeurs Chadia Moghrabi et Philippe Fournier-Viger, de leur encadrement et de leur dévouement – bien au-delà de leur devoir – tout au long de mes études. Leurs encouragements continus et leur soutien financier m'ont permis de mener à terme ce projet. Je leur suis très reconnaissant des innombrables heures qu'ils m'ont accordées et de la patience dont ils ont fait preuve. J'ai également apprécié leur souci du détail, leur enthousiasme, ainsi que leurs précieux conseils et commentaires, qui m'ont appris énormément.

Merci aussi aux membres du jury, Dr Bruno Emond et Dr Guillaume Durand, d'avoir évalué cette thèse. Leur expertise et leurs judicieux conseils m'ont aidé à peaufiner et valider ce travail. Merci également au professeur Éric Hervet d'avoir accepté de présider le jury.

Je remercie aussi les autres professeurs du Département d'informatique de leur encouragement et de leur enseignement. Je remercie particulièrement les professeurs et étudiants qui ont contribué à ce travail en acceptant généreusement de participer aux expériences.

Enfin, j'exprime ma profonde gratitude à mes parents, qui m'ont épaulé et encouragé tout au long de mes études universitaires. Je les remercie grandement de leur soutien moral et financier pendant toutes ces années.

TABLE DES MATIÈRES

RÉSUMÉ xi

RÉSUMÉ (anglais) xii

AVANT-PROPOS xiii

INTRODUCTION GÉNÉRALE 1

CHAPITRE I
CORRECTION AUTOMATIQUE DES QUESTIONS OUVERTES PAR LE BIAIS DES ONTOLOGIES DU WEB SÉMANTIQUE ET DES CONCEPTS FONCTIONNELS . . . 10

1 Introduction 11

2 Travaux antérieurs 14

3 Méthodologie 16

3.1 Représentation des connaissances du domaine 16

3.1.1 Ajout des concepts fonctionnels 16

3.1.2 Ordonnancement des concepts fonctionnels 19

3.2 Traitement des langues naturelles 21

3.3 Évaluation 21

3.3.1 Évaluation des connaissances conceptuelles 22

3.3.2 Évaluation des connaissances procédurales 23

4 Exemple pratique 24

5 Résultats expérimentaux 26

6 Conclusion 42

CHAPITRE II
ÉVALUATION DES CONNAISSANCES PROCÉDURALES DANS LES RÉPONSES EN TEXTE LIBRE PAR UNE APPROCHE HYBRIDE BASÉE SUR LE WEB SÉMANTIQUE 43

1 Introduction 44

2 Travaux antérieurs 46

3 Le système PROCMARK 48

3.1 Représentation des connaissances du domaine 49

3.1.1 Ontologie du cours 49

3.1.2 Réseau sémantique de la réponse idéale 49

3.2 Annotation des réponses des étudiants 57

3.2.1 Préparation des réponses 58

3.2.2 Recherche des correspondances 58

3.3 Création du réseau sémantique de l'étudiant 59
3.4 Évaluation des connaissances procédurales 63
3.4.1 Similarité du tout et partie 65
3.4.2 Proximité textuelle 66
3.4.3 Similarité lexicale 66
3.4.4 Proximité ontologique 67
3.4.5 Ordonnancement temporel 68
3.4.6 Évaluation 69
4 Résultats expérimentaux 71
5 Conclusion et travaux futurs 87
CONCLUSION GÉNÉRALE 89
ANNEXE A
ORDRES ATTENDUS POUR $\mathcal{DIJ}$, $\mathcal{DFS}$ ET $\mathcal{BT}$ 92
ANNEXE B
ONTOLOGIE DU COURS UTILISÉE PAR PROCMARK 94
ANNEXE C
QUESTIONS POSÉES AUX ÉTUDIANTS 99
ANNEXE D
RÉSEAU SÉMANTIQUE DE LA RÉPONSE IDÉALE POUR $\mathcal{DIJ}$ 101
ANNEXE E
RÉSEAU SÉMANTIQUE DE LA RÉPONSE IDÉALE POUR $\mathcal{DFS}$ 102
ANNEXE F
RÉSEAU SÉMANTIQUE DE LA RÉPONSE IDÉALE POUR $\mathcal{BT}$ 103
ANNEXE G
DICTIONNAIRE DE SYNONYMES (D1) UTILISÉ PAR PROCMARK 104
ANNEXE H
DICTIONNAIRE ONTOLOGIQUE (D2) UTILISÉ POUR $\mathcal{DIJ}$ 112
ANNEXE I
DICTIONNAIRE ONTOLOGIQUE (D2) UTILISÉ POUR $\mathcal{DFS}$ 117
ANNEXE J
DICTIONNAIRE ONTOLOGIQUE (D2) UTILISÉ POUR $\mathcal{BT}$ 119
RÉFÉRENCES 123

TABLE OF CONTENTS

ABSTRACT (French) xi

ABSTRACT xii

FOREWORD xiii

GENERAL INTRODUCTION 1

CHAPTER I
AUTOMATIC GRADING OF OPEN-ENDED QUESTIONS USING SEMANTIC WEB ONTOLOGIES AND FUNCTIONAL CONCEPTS 10

1 Introduction 11

2 Related Work 14

3 Methodology 16

3.1 Domain Knowledge Representation 16

3.1.1 Adding Functional Concepts 16

3.1.2 Ordering Functional Concepts 19

3.2 Natural Language Processing 21

3.3 Grading 21

3.3.1 Conceptual Grading 22

3.3.2 Functional Grading 23

4 Working Example 24

5 Experimental Results 26

6 Conclusion 42

CHAPTER II
ASSESSING PROCEDURAL KNOWLEDGE IN FREE-TEXT ANSWERS THROUGH A HYBRID SEMANTIC WEB APPROACH 43

1 Introduction 44

2 Literature Review 46

3 The PROCMARK System 48

3.1 Representing Domain Knowledge 49

3.1.1 Course Ontology 49

3.1.2 Semantic Network of Ideal Answer 49

3.2 Annotating Students' Answers 57

3.2.1 Preparation 58

3.2.2 Matching 58

3.3 Creating the Student's Semantic Network 59
3.4 Grading Procedural Knowledge 63
3.4.1 Part-Whole Similarity 65
3.4.2 Textual Proximity 66
3.4.3 Lexical Similarity 66
3.4.4 Ontological Proximity 67
3.4.5 Temporal Ordering 68
3.4.6 Grading 69
4 Experimental Results 71
5 Conclusion and Future Work 87
GENERAL CONCLUSION 89
APPENDIX A
EXPECTED ORDERINGS FOR $\mathcal{DIJ}$, $\mathcal{DFS}$, AND $\mathcal{BT}$ 92
APPENDIX B
COURSE ONTOLOGY USED BY PROCMARK 94
APPENDIX C
EXAM QUESTIONS 99
APPENDIX D
SEMANTIC NETWORK OF IDEAL ANSWER FOR $\mathcal{DIJ}$ 101
APPENDIX E
SEMANTIC NETWORK OF IDEAL ANSWER FOR $\mathcal{DFS}$ 102
APPENDIX F
SEMANTIC NETWORK OF IDEAL ANSWER FOR $\mathcal{BT}$ 103
APPENDIX G
THESAURUS (D1) USED BY PROCMARK 104
APPENDIX H
ONTOLOGICAL DICTIONARY (D2) USED FOR $\mathcal{DIJ}$ 112
APPENDIX I
ONTOLOGICAL DICTIONARY (D2) USED FOR $\mathcal{DFS}$ 117
APPENDIX J
ONTOLOGICAL DICTIONARY (D2) USED FOR $\mathcal{BT}$ 119
REFERENCES 123

LISTE DES TABLEAUX

I-1 Bloom's taxonomy and example verbs 12

I-2 Knowledge Dimension of the revised Bloom's taxonomy 14

I-3 Example ideal answer to describe Depth-First Search 21

I-4 Example annotated answer of a student to describe Depth-First Search 25

I-5 Annotated answer of a student to describe Depth-First Search, after inversion 25

I-6 Relative weights of concepts for $\mathcal{DFS}$ 28

I-7 Relative weights of concepts for $\mathcal{DIJ}$ 28

I-8 Relative weights of concepts for $\mathcal{BT}$ 28

I-9 Grades given by human graders for $\mathcal{DIJ}$ 29

I-10 Comparison of average human grades with those of OeLE and our system for $\mathcal{DIJ}$. 30

I-11 Ordering factor of student answers and average human grades for $\mathcal{DIJ}$ 32

I-12 Grades given by human graders for $\mathcal{DFS}$ 33

I-13 Comparison of average human grades with those of OeLE and our system for $\mathcal{DFS}$. 34

I-14 Ordering factor of student answers and average human grades for $\mathcal{DFS}$ 36

I-15 Grades given by human graders for $\mathcal{BT}$ 37

I-16 Comparison of average human grades with those of OeLE and our system for $\mathcal{BT}$ . . 39

I-17 Ordering factor of student answers and average human grades for $\mathcal{BT}$ 41

II-1 Categories of concepts and examples 51

II-2 Relations and example usage 54

II-3 Annotations of student's answer 61

II-4 Symbols table for example answer 63

II-5 Temporal ordering values for all answer permutations 70

II-6 Grades given by human graders for $\mathcal{DIJ}$ 72

II-7 Comparison of average human grades with those of PROCMARK for $\mathcal{DIJ}$ 73

II-8 Correlations between pairs of human graders and PROCMARK for $\mathcal{DIJ}$ 75

II-9 Grades given by human graders for $\mathcal{DFS}$ 78

II-10 Comparison of average human grades with those of PROCMARK for $\mathcal{DFS}$ 79

II-11 Correlations between pairs of human graders and PROCMARK for $\mathcal{DFS}$ 80

II-12 Grades given by human graders for $\mathcal{BT}$ 82

II-13 Comparison of average human grades with those of PROCMARK for $\mathcal{BT}$ 84

II-14 Correlations between pairs of human graders and PROCMARK for $\mathcal{BT}$ 86

LISTE DES FIGURES

1 Évaluation des compétences de l'apprenant par le biais de ses compétences 1

2 Évaluation des compétences par les systèmes informatiques 3

3 Réseau sémantique représentant la phrase « Le temps est beau en Bavière aujourd'hui. » dans le MultiNet Working Bench 7

I-1 Course ontology used for automatic assessment 27

I-2 Comparison of human grades with those of OeLE and our system for $\mathcal{DIJ}$ 31

I-3 Correlation of human grades with those of the system for $\mathcal{DIJ}$ 31

I-4 Comparison of human grades with those of OeLE and our system for $\mathcal{DFS}$ 35

I-5 Correlation of human grades with those of the system for $\mathcal{DFS}$ 38

I-6 Comparison of human grades with those of OeLE and our system for $\mathcal{BT}$ 40

I-7 Correlation of human grades with those of the system for $\mathcal{BT}$ 40

II-1 Partial concept class hierarchy 50

II-2 Example of attribute-concept and literal-concept in the semantic network 53

II-3 Example of relation-concepts and relations in the semantic network 55

II-4 Example of the *decomposition* and *operand* relations in the semantic network 56

II-5 Example of nesting *AndThen* and *OrThen* relation-concepts in the semantic network . 57

II-6 Example of a student's semantic network overlay after the Matching phase 59

II-7 The semantic network corresponding to the phrase "parent of W" 60

II-8 The semantic network corresponding to the phrase "V is visited", where *EqualTo* operator concept and *True* literal concept are inferred 64

II-9 Example of partially matched subgraph in the semantic network 65

II-10 Example of temporal ordering with *AndThen* in the semantic network 68

II-11 Nested *AndThen* and *OrThen* relation-concepts in the semantic network 69

II-12 Comparison of human grades with those of PROCMARK for $\mathcal{DIJ}$ 74

II-13 Correlation of human grades with those of PROCMARK for $\mathcal{DIJ}$ 75

II-14 Ranking of human grades compared to PROCMARK for $\mathcal{DIJ}$ 76
II-15 Correlation of individual measures for $\mathcal{DIJ}$ 76
II-16 Comparison of human grades with those of PROCMARK for $\mathcal{DFS}$ 77
II-17 Correlation of human grades with those of PROCMARK for $\mathcal{DFS}$ 80
II-18 Ranking of human grades compared to PROCMARK for $\mathcal{DFS}$ 81
II-19 Correlation of individual measures for $\mathcal{DFS}$ 83
II-20 Comparison of human grades with those of PROCMARK for $\mathcal{BT}$ 85
II-21 Correlation of human grades with those of PROCMARK for $\mathcal{BT}$ 85
II-22 Ranking of human grades compared to PROCMARK for $\mathcal{BT}$ 86
II-23 Correlation of individual measures for $\mathcal{BT}$ 87

RÉSUMÉ

La venue du World Wide Web, dans les années 1990, a entraîné un intérêt croissant pour l'apprentissage en ligne. De nombreux collèges et universités offrent des cours en ligne menant à des diplômes et autres certifications. Un aspect pragmatique de l'apprentissage en ligne a retenu notre attention : celui de la correction automatique des réponses en texte libre des étudiants aux épreuves. La correction automatique peut aussi être appliquée à l'évaluation des dissertations reçues – en grand nombre – par certaines universités comme critère d'admission ou, de façon plus générale, à l'évaluation du contenu, de la structure, ou même du style des compositions des étudiants. Elle s'avère particulièrement utile à l'apprentissage en ligne, mais elle peut aussi être utilisée en classe. En plus d'être rapide, la correction automatique a l'avantage d'être objective, car l'intervention humaine est réduite.

Nous nous intéressons au problème particulier de la correction automatique des épreuves où les étudiants sont appelés à décrire une procédure ou une séquence de procédures. Ces connaissances, que nous appelons « connaissances procédurales », sont essentielles à plusieurs domaines, et cruciales en informatique. La correction devient d'autant plus ardue que les professeurs et correcteurs doivent non seulement vérifier l'exactitude des faits ou concepts énoncés par l'étudiant, mais aussi l'ordre dans lequel ils sont énoncés.

Ce travail présente deux versions de notre système, PROCMARK, conçu et développé pour l'évaluation des connaissances procédurales. Notre approche consiste à analyser le contenu syntaxique et sémantique des réponses, en employant des techniques du Web sémantique et de traitement des langues naturelles, tout en tenant compte de la séquence de procédures attendue dans les réponses. Les résultats de notre système sont semblables à ceux des correcteurs humains, avec des corrélations entre 0,59 et 0,86.

Mots clés : évaluation automatique ; connaissances procédurales ; apprentissage en ligne

ABSTRACT

The advent of the World Wide Web, in the 1990s, brought a growing interest towards e-learning. Multiple colleges and universities offer online courses which can lead to diplomas and other certifications. A pragmatic aspect of e-learning captured our attention: the automatic assessment of free-text answers to exam questions. Automatic assessment can also assist in the evaluation of essays received – in high numbers – by some universities as part of the admission process. It can also be used for the evaluation of the content, structure, or even style of student essays. Automatic assessment proves especially useful in an online course environment, although it can also be applied to classroom teaching. It is not only faster, but also more objective than manual scoring, since human intervention is kept to a minimum.

We address the particular challenge of automatic assessment of student texts describing a procedure or a sequence of procedures. This type of knowledge, which we call "procedural knowledge", is essential to many domains, and crucial in Computer Science. When assessing such knowledge, professors and graders need to verify both the facts and concepts expressed in the student answer, as well as the order in which they appear in the text.

We present two versions of our system, PROCMARK, which was specifically designed and developed for the automatic assessment of procedural knowledge. Our approach relies on Semantic Web and Natural Language Processing techniques to analyze the syntactic and semantic content of student answers, while taking into consideration the expected sequence of procedures in the answers. Our system gives grades which are similar to those of human graders, with correlations ranging from 0.59 to 0.86.

Keywords: computer-assisted assessment; procedural knowledge; e-learning

AVANT-PROPOS

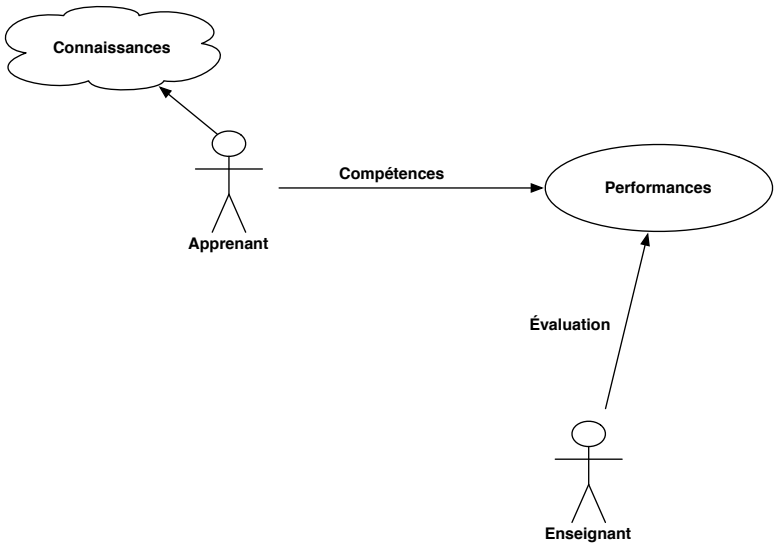
Cette thèse est divisée en deux chapitres, qui correspondent à trois de nos publications. Le premier chapitre est une version augmentée de nos publications [1, 2]. La première, intitulée « Assessing Procedural Knowledge in Open-ended Questions through Semantic Web Ontologies », a été publiée dans les actes de la conférence *AOW2012 : 8th Australasian Ontology Workshop, in conjunction with the 25th Australasian Joint Conference on Artificial Intelligence*, tenue à Sydney, Australie, en décembre 2012. La deuxième, intitulée « Automatic Grading of Open-ended Questions Using Semantic Web Ontologies and Functional Concepts », sera publiée par Springer et paraîtra en 2015 dans le livre *Ontologies : Theory and Applications in Information Systems and the Semantic Web*. Cette partie du travail présente notre recherche bibliographique, ainsi que la conception et le développement d'une application Web permettant l'évaluation automatique des connaissances procédurales.

Le second chapitre est une version augmentée de notre article [3] intitulé « Assessing Procedural Knowledge in Free-text Answers through a Hybrid Semantic Web Approach », publié dans les actes de la conférence *ICTAI 2013: 25th IEEE International Conference on Tools with Artificial Intelligence*, tenue du 4 au 6 novembre 2013 à Washington, D.C. Ce travail présente notre système, PROCMARK, conçu et développé pour l'évaluation des connaissances procédurales. Nous discutons de nos travaux envers la formalisation d'une ontologie procédurale, et faisons état de nos résultats améliorés, grâce aux nouvelles mesures de similarité que nous avons exploitées. Nous combinons l'approche décrite dans le premier chapitre aux réseaux sémantiques dans le but de simplifier la représentation de la réponse idéale et de faciliter l'annotation des réponses des étudiants.

INTRODUCTION GÉNÉRALE

L'évaluation de l'apprentissage des étudiants est certes une tâche incontournable et essentielle. Toutefois, bien que les professeurs puissent corriger rapidement et sans grande difficulté les questions fermées, à choix multiples ou de type vrai ou faux, ou encore à réponse courte, l'évaluation des questions d'analyse ou de synthèse (les questions ouvertes, en texte libre, ou « à développement ») devient souvent exigeante et fastidieuse. De leur côté, les étudiants redoutent parfois la subjectivité du processus, qui est cependant inévitable de par la nature même des langues naturelles et des communications interpersonnelles. Selon [4], l'enseignant n'évalue pas directement les connaissances des apprenants, mais plutôt leurs compétences, par le biais de leurs performances qui sont directement observables, tel que démontré dans la Figure 1.

FIGURE 1 Évaluation des compétences de l'apprenant par le biais de ses compétences



L'évaluation par ordinateur se prête particulièrement bien à l'évaluation dans un environnement d'apprentissage en ligne, bien qu'elle puisse également être appliquée à l'apprentissage en classe. La correction des questions fermées ou à réponse courte est triviale pour l'ordinateur [1] ; mais l'évaluation des réponses en texte libre s'avère plus difficile. Il n'est donc pas surprenant que

1. La correction des questions à choix multiples peut même être automatisée par l'utilisation d'un lecteur optique.

l'évaluation automatique de textes courts fasse l'objet de nombreuses recherches. Elle peut en outre servir à évaluer les dissertations reçues – en grand nombre – par certaines universités comme critère d'admission ou, de façon plus générale, à l'évaluation du contenu, de la structure, ou même du style des compositions des étudiants. En plus d'être rapide, l'évaluation automatique a l'avantage d'être objective, car l'intervention humaine est réduite.

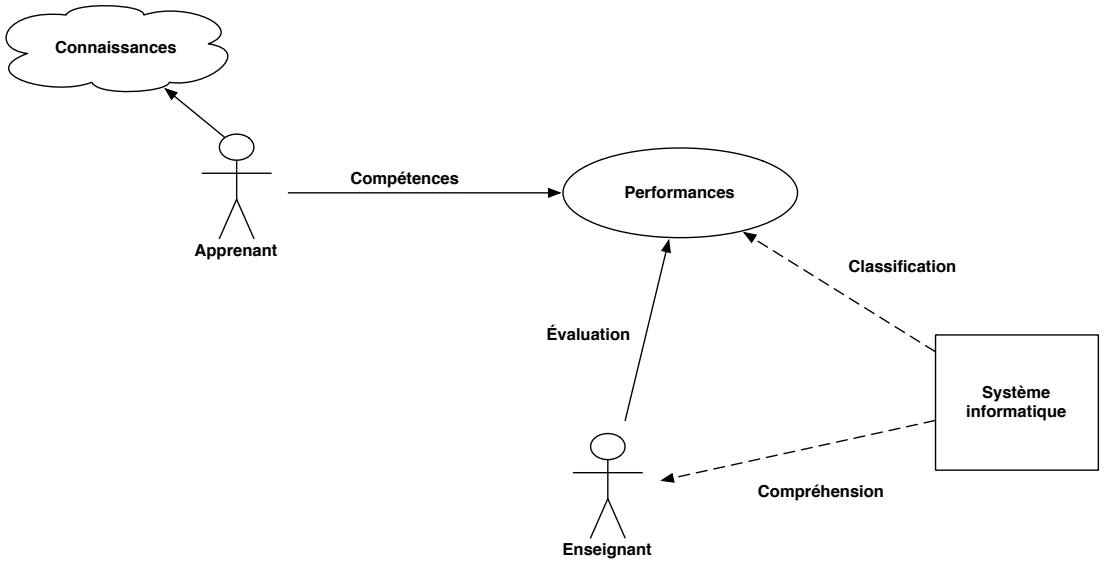
La recherche a donné lieu à plusieurs systèmes, qui utilisent différentes techniques, comme l'extraction d'information [11, 12], le traitement des langues naturelles [13–18], ou des techniques statistiques [20–22], selon le problème particulier qu'ils tentent de résoudre. On peut généralement séparer les systèmes en deux catégories. Les premiers effectuent la *classification* des nouveaux textes en les comparant à un ensemble de textes d'entraînement, à des réponses d'étudiants précédemment notées, ou encore à des réponses de référence fournies par le professeur. Les seconds visent la *compréhension* plus profonde des textes par une analyse syntaxique ou sémantique du contenu, en y appliquant des techniques de traitement des langues naturelles [2].

La Figure 2 illustre comment les systèmes informatiques qui effectuent la classification observent directement les performances des apprenants, tandis que les systèmes qui visent la compréhension tentent de reproduire la façon dont l'enseignant évalue l'apprentissage des apprenants. Dans tous les cas, les résultats expérimentaux confirment surtout l'accord [3] des notes données par les systèmes avec celles des humains, et non la qualité de l'évaluation elle-même. On parle en outre ici d'évaluation sommative et normative [4].

Deux systèmes visant la compréhension plus profonde des textes retiennent notre attention. Le premier, OeLE [8], est issu de la Universidad de Murcia (Espagne). Ce système se distingue non seulement par son analyse sémantique des textes des étudiants, mais aussi par son appel aux technologies du Web sémantique pour la représentation du matériel didactique, en particulier par des ontologies encodées dans des fichiers [4]. Une ontologie, au sens informatique du terme, est une représentation formelle des connaissances par le biais d'un schéma exprimant le contenu sémantique. Les concepts du domaine étudié sont reliés par des relations taxinomiques et sémantiques. De plus,

2. *Natural Language Processing*, ou NLP.
3. *Agreement*, en anglais, ou encore la corrélation. Voir les Sections 2 des deux chapitres.
4. Par exemple, en utilisant le *Resource Description Framework* (RDF) ou le *Web Ontology Language* (OWL).

FIGURE 2 Évaluation des compétences par les systèmes informatiques



les concepts peuvent être qualifiés par des attributs. La création d'une ontologie est donc une tâche qui requiert à la fois des compétences techniques et une connaissance approfondie du domaine concerné. Le Web sémantique promet toutefois le partage et la réutilisation des ontologies.

OeLE utilise une réponse idéale fournie par le professeur du cours pour évaluer semiautomatiquement les réponses des étudiants. Une interface graphique permet au professeur d'annoter sa propre réponse ainsi que celles des étudiants à partir des entités (concepts, relations et attributs) présentes dans l'ontologie. L'annotation consiste essentiellement à associer aux entités abstraites de l'ontologie les termes concrets (c'est-à-dire, les mots ou expressions) utilisés dans les textes pour les représenter, formant ainsi une base de connaissances [5]. L'annotation de chaque nouvelle réponse vient enrichir la base de connaissances ; de ce fait, le professeur n'a qu'à annoter les expressions inconnues. Ainsi, il est concevable que le processus d'évaluation devienne complètement automatique, si le corpus est suffisamment grand.

Le deuxième système, MultiNet Working Bench (MWB) [19], développé à la FernUniversität Hagen (Allemagne), vise également la compréhension des textes des étudiants par le biais de réseaux sémantiques. Un outil graphique permet la représentation visuelle du réseau sémantique ex-

5. *Knowledge Base*, ou KB.

trait de la réponse de l'étudiant. Ce réseau est ensuite comparé au réseau de référence soumis par le professeur. Les parties qui correspondent sont correctes et sont affichées en vert, alors que les autres sont affichées en rouge.

Dans le cadre de ce travail, nous nous intéressons à l'évaluation automatique des réponses en texte libre des étudiants à des questions d'examen, dans le contexte particulier des épreuves visant l'évaluation des connaissances procédurales. Nous définissons ici les connaissances procédurales comme les connaissances permettant de *décrire* les procédures ou les tâches. Anderson et Krathwohl [9] les définissent plutôt comme les connaissances permettant *d'accomplir* une tâche (savoir-faire). Notre définition des connaissances *procédurales* est donc un cas particulier de leur définition des connaissances *conceptuelles*, car celle-ci regroupe "les connaissances des relations entre les éléments de base (connaissances factuelles) au sein d'une plus grande structure". Plus précisément, nos connaissances procédurales sont donc un ensemble partiellement ou totalement ordonné de connaissances factuelles et conceptuelles.

Notre définition s'apparente donc plus aux réseaux de tâches hiérarchiques [6] [6], issus des travaux sur la planification en intelligence artificielle. Dans ces réseaux, les tâches composées peuvent être décomposées en sous-tâches, et les tâches primitives sont celles qui peuvent être accomplies directement. Nos connaissances procédurales sont essentiellement récursives, au sens où chacune des étapes qui les composent peut être à son tour décomposée en connaissances procédurales, pour donner une description plus ou moins détaillée d'un procédé.

Les connaissances procédurales sont cruciales à plusieurs domaines, comme par exemple les cours de premiers soins, le soutien technique téléphonique, ou encore des techniques d'assemblage qui demandent d'effectuer des tâches dans un ordre précis. Bien que les systèmes existants accomplissent avec succès l'évaluation des connaissances factuelles et conceptuelles dans les réponses des étudiants, nous observons que les connaissances procédurales ne sont pas prises en compte. Par exemple, en informatique, trois étapes (ou instructions) sont requises pour échanger le contenu de deux variables. Il est non seulement crucial que les trois étapes apparaissent dans la réponse de l'étudiant, mais aussi dans un ordre déterminé.

---

6. *Hierarchical Task Network*, ou HTN.

Plusieurs systèmes informatiques visent particulièrement l'évaluation des compétences des étudiants dans les domaines de l'informatique [11, 19, 21, 22, 31]. Le lecteur est invité à consulter [5] pour une bonne revue de ce type de systèmes. Dans le cadre de notre travail, nous nous intéressons à l'évaluation des connaissances procédurales par l'analyse de textes libres écrits en langue naturelle (c'est-à-dire en français), et non en code ou en pseudocode. De cette façon, il serait concevable d'utiliser notre système pour l'évaluation de connaissances procédurales dans d'autres domaines que l'informatique.

Le présent travail est divisé en deux chapitres. Le premier chapitre présente notre première approche, qui enrichit OeLE pour permettre l'évaluation des connaissances procédurales à l'aide de concepts fonctionnels. Les connaissances factuelles et conceptuelles sont ici aussi représentées dans une ontologie. L'ontologie est encodée dans le Web Ontology Language (OWL), ce qui permet de la partager avec d'autres professeurs et de la réutiliser par l'entremise du Web sémantique. La nouveauté de notre approche réside dans l'ajout de cette nouvelle catégorie de concepts à l'ontologie, les *concepts fonctionnels*.

Les concepts fonctionnels s'apparentent aux fonctions et procédures de l'informatique. Ils représentent une procédure (ou une tâche), une séquence de sous-procédures (ou sous-tâches), ou les étapes individuelles nécessaires à l'accomplissement d'une tâche. Les concepts fonctionnels peuvent être imbriqués, de la même façon qu'une procédure peut faire appel à plusieurs sous-procédures. De plus, les concepts fonctionnels peuvent être réutilisés (ou plus précisément, plusieurs instances du même concept fonctionnel peuvent être déclarées) pour simuler l'appel d'une même sous-procédure par plusieurs procédures.

Une méta-ontologie permet ensuite d'ordonner un sous-ensemble de concepts fonctionnels ; autrement dit, le sous-ensemble de concepts qui doivent apparaître dans un certain ordre dans les réponses des étudiants. Une relation temporelle transitive est introduite à cet effet ; chaque instance de la relation permet d'indiquer qu'un concept fonctionnel doit en suivre un autre. Notre approche tient compte de ces relations temporelles dans le calcul de la note de l'étudiant. L'étudiant est pénalisé chaque fois qu'une relation d'ordre n'est pas respectée.

Le second chapitre présente notre système, PROCMARK, qui amène plusieurs nouveautés. Dans cette nouvelle version, nous formalisons davantage l'ontologie en l'augmentant de notions d'algorithmique plus générales. Aux concepts fonctionnels, nous ajoutons 10 catégories de concepts, qui représentent les structures de contrôle, les conditions, les variables, etc.

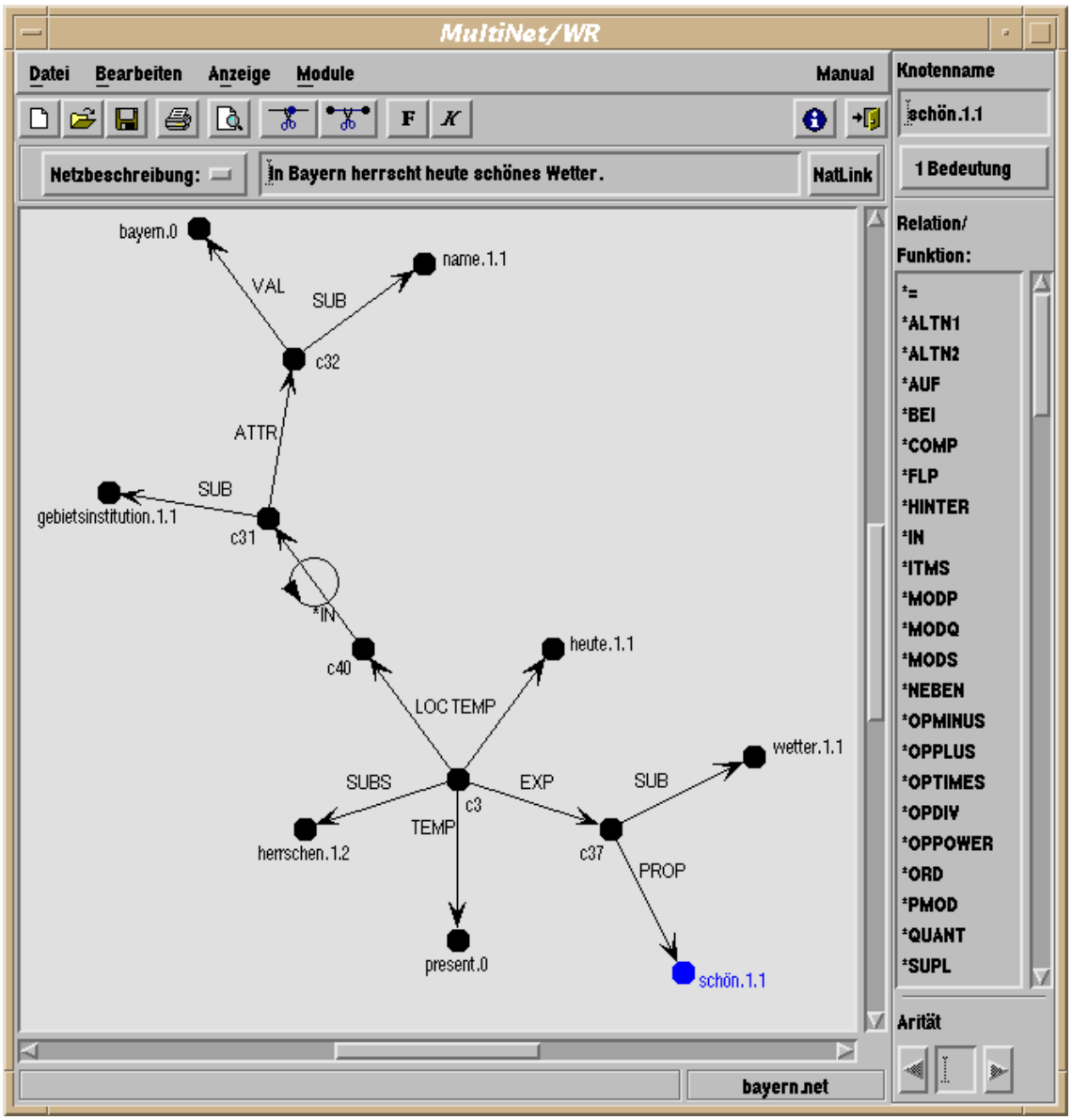
Dans le but de simplifier la création de l'ontologie, tout en tenant compte des contraintes de OWL, nous paramétrons les relations, les attributs et les littéraux comme des concepts [7], pour devenir des *relations-concepts*, des *attributs-concepts* et des *littéraux-concepts*. Notre ontologie algorithmique devient ainsi plus flexible en permettant aux relations-concepts d'être imbriquées, c'est-à-dire de prendre d'autres relations-concepts comme domaine ou image. Les relations en OWL ne prennent que des concepts comme domaine ou image, ce qui rend difficile, par exemple, la création d'un arbre d'expressions.

Notons ici que la granularité de l'ontologie se trouve considérablement augmentée par rapport à notre première approche, ce qui simplifie l'annotation. En effet, on peut avancer que plus les concepts sont précis, moins il existe de variations linguistiques pour les exprimer. En maximisant la décomposition des concepts jusqu'à l'obtention de concepts « atomiques » (ou indivisibles), nous minimisons l'annotation manuelle, et déléguons au système la tâche de recomposer les concepts plus généraux à partir des concepts atomiques.

En nous inspirant du MultiNet Working Bench [19], nous choisissons de représenter la réponse idéale fournie par le professeur par un réseau sémantique. Les noeuds de ce réseau sont des instances des concepts de l'ontologie et les arcs, des instances des relations [8]. La Figure 3 montre un réseau sémantique dans le MultiNet Working Bench. MultiNet est décrit comme un « formalisme expressif permettant une représentation très détaillée du sens ». Nous définissons 15 relations dans l'ontologie, qui facilitent la création du réseau sémantique. Parmi ces relations, notons la relation *decomposition*, qui formalise le développement d'une procédure en sous-procédures. Les relations *doBefore* et *doAfter* viennent remplacer la relation temporelle de notre première approche, et sont combinées avec les concepts représentant les structures de contrôle afin d'indiquer la séquence souhaitée.

7. Autrement dit, par réification, les relations, les attributs et les littéraux deviennent « citoyens de première classe ».
8. Et non des relations-concepts.

FIGURE 3 Réseau sémantique représentant la phrase « Le temps est beau en Bavière aujourd'hui. » dans le MultiNet Working Bench



Lors de l'évaluation, les réponses des étudiants sont transformées en réseaux sémantiques, qui sont superposés au réseau sémantique de la réponse idéale. Les concepts fonctionnels peuvent être décomposés en divers degrés de granularité dans la réponse idéale, selon le niveau de détail attendu des réponses des étudiants. Un processus d'annotation semblable à celui de [8] permet d'associer les expressions linguistiques présentes dans les réponses des étudiants aux concepts de l'ontologie par le biais d'un dictionnaire. Notre approche est cependant plus flexible, puisque les relations, attributs et littéraux (devenus relations-concepts, attributs-concepts et littéraux-concepts) sont annotés uniformément. À partir des expressions linguistiques qui y correspondent, notre système détermine la permutation qui minimise la distance entre les expressions. Les réseaux sémantiques des réponses servent ainsi en quelque sorte d'arbre syntaxique, et nous dispensent des motifs grammaticaux, de sorte que le système puisse facilement être adapté à d'autres langues en remplaçant simplement le dictionnaire.

Enfin, nous améliorons l'algorithme d'évaluation de notre première approche en y ajoutant trois nouvelles mesures de similarité. La première, *part-whole similarity* (similarité du tout et partie), permet de comparer le niveau de détail fourni par l'étudiant à celui attendu par le professeur. Cette mesure évalue donc la proportion de la réponse idéale trouvée dans la réponse d'un étudiant.

La seconde mesure, *textual proximity* (proximité textuelle), s'inspire du logiciel ALA-Mapper [29], développé à la Penn State University (États-Unis), qui permet de vérifier par une interface graphique les schémas conceptuels [9] créés par les étudiants. Ce logiciel définit la proximité entre deux concepts en termes de coordonnées (en pixels) sur l'écran. Puisque notre système utilise des données textuelles, nous définissons la proximité textuelle entre un concept et les concepts qui y sont reliés à partir du nombre de mots qui les séparent dans le texte.

Enfin, notre algorithme évalue le *temporal ordering* (ordonnancement temporel). Cette mesure considère l'ordre relatif (en termes de position dans le texte) des concepts fonctionnels dans les réponses des étudiants, toujours par rapport à la réponse idéale fournie par le professeur. Les instances des concepts représentant les structures de contrôle, combinées aux instances des relations *doBefore* et *doAfter*, sont utilisées pour le calcul, ce qui nous évite de recourir à une méta-ontologie.

9. *Concept maps*, ou *cognitive maps*.

Nous comparons l'évaluation automatique de nos systèmes à l'évaluation manuelle (par plusieurs correcteurs) de réponses d'étudiants à trois questions d'examen dans des cours portant sur des algorithmes informatiques offerts à l'Université de Moncton entre 2011 et 2013. Au total, 53 étudiants et 7 correcteurs humains ont participé à nos expériences. Les étudiants sont appelés à décrire le fonctionnement de deux algorithmes de recherche dans un graphe : la recherche en profondeur, ainsi que la recherche du plus court chemin (Dijkstra). La troisième question consiste à expliquer l'algorithme de recherche d'une valeur dans un arbre B+.

Comme seule consigne, les étudiants sont invités à répondre par des phrases simples. Pour leur part, les correcteurs humains ne reçoivent aucune consigne, sauf d'attribuer une note numérique aux réponses des étudiants selon leurs exigences et critères habituels. Afin de ne pas fausser les résultats, les correcteurs humains n'ont pas eu accès aux notes accordées par le système, et les noms des étudiants ne leur ont pas été divulgués. Les professeurs connaissaient cependant les noms de leurs propres étudiants, car ils ont évalué les épreuves dans le cadre normal d'un cours.

À l'instar de d'autres systèmes [21, 22], nous comparons les notes données par le système à celles données par les humains en utilisant la moyenne, l'écart-type et la corrélation de Pearson. Nous avons effectué l'analyse de régression afin de déterminer la signification statistique. Nous prévoyons employer des analyses statistiques plus avancées pour la comparaison, telles que la corrélation intraclasse [10] dans nos travaux futurs.

10. *Intraclass Correlation Coefficient*, ou ICC.

CHAPITRE I

AUTOMATIC GRADING OF OPEN-ENDED QUESTIONS USING SEMANTIC WEB ONTOLOGIES AND FUNCTIONAL CONCEPTS

CORRECTION AUTOMATIQUE DES QUESTIONS OUVERTES PAR LE BIAIS DES ONTOLOGIES DU WEB SÉMANTIQUE ET DES CONCEPTS FONCTIONNELS

Abstract

This paper presents a novel approach for the automatic assessment of students' answers to open-ended questions. It improves on the OeLE method, which uses ontologies and Semantic Web technologies to represent course material. The main novelty of our approach is the addition of a new category of concepts, named *functional concepts*, which can be temporally ordered. This modification allows the assessment of procedural knowledge in students' answers by taking the ordering of these concepts into account. We present an example and experimental results for the grading of answers obtained from three groups of a computer algorithms course. Results show a positive correlation (0.59, 0.67, and 0.86) between the system's scores and those of human graders.

Keywords: e-learning; computer-assisted assessment; ontology; Semantic Web; procedural knowledge

1 Introduction

Grading student's papers by hand can be tedious and time-consuming for instructors. Several early systems attempted to automate such evaluation. However, assessing the students' learning in an e-learning environment often relies on multiple choice or fill-in-the-blank questions, which only trigger the lowest level (*Knowledge*) of Bloom's taxonomy of knowledge acquisition [7]. As we shall see in Section 2, several attempts have been made to incorporate open-ended questions in online assessments, which could possibly trigger the higher levels of Bloom's taxonomy (*Synthesis* and *Evaluation*) in the students' learning. Table I-1 shows the six levels of Bloom's taxonomy, along with example verbs that can appear in exam questions.

To build an e-learning environment that can automatically grade free-text answers, a variety of techniques have been used, such as Information Extraction (IE) [11, 12], Natural Language Processing (NLP) [13–18], or statistical techniques [20–22]. Automatic assessment is faster, and is performed more objectively than manual scoring. Furthermore, [10] distinguishes between shallow and full NLP techniques. Shallow techniques refer to a more superficial analysis of the text, using less complex NLP to compare the text to a training set with statistical techniques. Full NLP

Table I-1 Bloom's taxonomy and example verbs

Level	**Example verbs**
Knowledge	Count, Define, Describe, Draw, Find, Identify, Label, List, Match, Name, Quote, Recall, Recite, Sequence, Tell, Write
Comprehension	Conclude, Demonstrate, Discuss, Explain, Generalize, Identify, Illustrate, Interpret, Paraphrase, Predict, Report, Restate, Review, Summarize, Tell
Application	Apply, Change, Choose, Compute, Dramatize, Interview, Prepare, Produce, Role-play, Select, Show, Transfer, Use
Analysis	Analyze, Characterize, Classify, Compare, Contrast, Debate, Deduce, Diagram, Differentiate, Discriminate, Distinguish, Examine, Outline, Relate, Research, Separate
Synthesis	Compose, Construct, Create, Design, Develop, Integrate, Invent, Make, Organize, Perform, Plan, Produce, Propose, Rewrite
Evaluation	Appraise, Argue, Assess, Choose, Conclude, Critic, Decide, Evaluate, Judge, Justify, Predict, Prioritize, Prove, Rank, Rate, Select

techniques imply a more complex analysis of the text, by identifying entities and syntactic dependencies with respect to an ontology expressing the course material in a machine-readable format. Among the systems discussed in Section 2, [16, 17, 19] use Full NLP techniques.

Our approach extends the OeLE system [8], which also uses Full NLP techniques to assess the level of understanding of the students. Its course material is represented in an ontology encoded in the Web Ontology Language (OWL). The system has been used in two online courses, *Design and Evaluation of Didactic Media*, and *Multimedia Systems and Graphical Interaction*. It successfully assesses the semantic content of the students' answers containing factual expressions about didactic media or multimedia systems. However, when applying it to domains where procedural knowledge is predominant, such as computer algorithms, we observed that the ordering of the elements in the students' answers was not taken into account.

Table I-2 shows the *Knowledge Dimension* of Anderson and Krathwohl's revision [9] of Bloom's taxonomy. They define procedural knowledge as "the knowledge of subject-specific skills and algorithms"; that is, "how to do something". We define *procedural knowledge* as knowledge *about* procedures; or, for computer algorithms, "how to instruct a computer to do something". Our definition is thus a special case of conceptual knowledge, where the interrelationships among the basic elements (computer instructions) are assembled within a larger structure (computer program) in a specific order.

It is crucial that this ordering be considered when dealing with algorithms assessment. We address this challenge through a new approach, which introduces *functional concepts*. A meta-ontology incorporates ordering information about the subset of functional concepts that need to be ordered. The novelty of our work resides in considering the order of these concepts in the students' answers and adjusting their grade accordingly. Moreover, the use of Semantic Web technologies allows sharing and reusing course ontologies, thus reducing the time spent designing course material. Semantic techniques promise a deeper understanding of the text than statistical techniques.

Section 2 of this paper is a review of automatic free-text assessment systems. Section 3 presents the general methodology, followed by a working example in Section 4. We present our experimental results in Section 5, and conclude the paper in Section 6.

Table I-2 Knowledge Dimension of the revised Bloom's taxonomy

Level	Description
Factual Knowledge	The basic elements that students must know to be acquainted with a discipline or solve problems in it.
Conceptual Knowledge	The interrelationships among the basic elements within a larger structure that enable them to function together.
Procedural Knowledge	How to do something; methods of inquiry, and criteria for using skills, algorithms, techniques, and methods.
Metacognitive Knowledge	Knowledge of cognition in general as well as awareness and knowledge of one's own cognition.

2 Related Work

This section presents previous and ongoing research in automatic short-answer assessment. We only focus here on short-answer assessment systems, where reference texts are tailored to the course material, although some other systems have also been developed for essay scoring, where more general texts about a topic are used for training. A good review of many of these systems can be found in [10].

Some systems compare students' answers to an ideal answer supplied by the teacher. For instance, Automated Text Marker [11] uses a pattern-matching technique. It was tested in courses on Prolog programming, psychology and biology. Automark [12] uses Information Extraction techniques to grade students' answers by comparing them to a mark scheme template provided by the teacher. It achieved 94.7% agreement with human grading for six of the seven science-related questions asked on a test exam.

Some systems require teachers to provide training sets of marked student answers. For example, Auto-marking [13] uses NLP and pattern-matching techniques to compare students' answers to a training set of marked student answers. This system obtained 88% of exact agreement with human grading in a biology course. Bayesian Essay Test Scoring SYstem (BETSY) [14] uses naive Bayes

classifiers to search for specific features in students' answers. In a biology course, it achieved up to 80% accuracy. CarmelTC [15] uses syntactic analysis and naive Bayes classifiers to analyze essay answers. In an experiment with 126 physics essays, it obtained 90% precision and 80% recall. The Paperless-School Marking Engine (PS-ME) [16] requires a training set of marked answers and uses NLP to grade the answers. It incorporates Bloom's taxonomy heuristics; however, it is a commercial product and its exact implementation is not disclosed. C-rater [17] uses a set of marked training essays to determine the students' grade using NLP. In an experiment of 170,000 answers to reading comprehension and algebra questions, it achieved 85% accuracy. In [18], a combination of NLP and Support Vector Machines is used to classify answers into two classes (above/below 6 points out of 10). It obtained an average of 65% precision.

The MultiNet Working Bench system [19] uses a graphical tool to represent the students' knowledge visually. It compares the student's semantic network to that submitted by the teacher. Verified parts of the network are displayed in green, while wrong or unverified parts (not supported by logic inference) are displayed in red.

Other systems rely on Latent Semantic Analysis (LSA). For example, Research Methods Tutor [20] compares the students' answers to a set of expected answers. If the student answers incorrectly, the system guides the student into obtaining the right answer. The Willow system [21] requires several unmarked reference answers for each question. It also uses LSA to evaluate students' answers written in English or Spanish. In a computer science course, it achieved on average 0.54 correlation with the teacher's grading. A system currently in use at the University of Witwatersrand [22] uses LSA and clustering techniques. It achieves between 0.80 and 0.95 correlation with the teacher's grading.

To our knowledge, no system is specifically designed to assess procedural knowledge in open-ended answers, although such a system would be helpful in an e-learning environment for the numerous scientific and technical domains.

3 Methodology

In this section, we present our system, which comprises three phases: domain knowledge representation, natural language processing (NLP), and grading. We adapt the OeLE approach, expand it to the assessment of procedural knowledge, and use it to grade students' answers obtained from two groups of a computer algorithms course taught in French at Université de Moncton.

For each of the exam questions, the student's answers are compared to an ideal textual answer provided by the teacher. An in-house system performs the NLP tasks, such as tokenization, stemming, and synonym resolution, on both the students' and teacher's answers. The Protégé [1] software is used to build the course ontologies and encode them in OWL. Our system is written in PHP, which is well-suited to the development of a Web application relying heavily on text processing. We also developed our own ontology-processing code – which amounts mostly to graph traversal – to avoid relying on a full-fledged framework like Jena. Below, we refer to OWL classes as *concepts*, to object properties as *relations*, and to data properties as *attributes*.

3.1 Domain Knowledge Representation

The first phase consists of defining the course ontology to represent the procedural knowledge, from which the ideal answer will be constructed. In order to assess procedural knowledge, our system takes into account the relative order of a subset of concepts expressing procedural knowledge. Our system expands on the OeLE system, which appears to be designed for conceptual knowledge assessment, by adding a special category of concepts, *functional concepts*, which can be temporally ordered.

3.1.1 Adding Functional Concepts

In order to distinguish concepts that express procedural knowledge from those expressing factual or conceptual knowledge, we propose the addition of *functional concepts* to the course ontology. A functional concept logically represents a global procedure, a sequence of sub-procedures,

1. http://protege.stanford.edu/

or individual steps needed to accomplish a given task. Let us consider the following example algorithm, *DepthFirstSearch*, given here in pseudocode:

```
procedure DepthFirstSearch
    VisitRoot
    VisitFirstChildNode
    VisitOtherSiblings
end
procedure VisitRoot
    [...]
end
procedure VisitFirstChildNode
    [...]
end
procedure VisitOtherSiblings
    [...]
end
```

For every procedure or sub-procedure, we create a corresponding functional concept: *DepthFirstSearch*, *VisitRoot*, *VisitFirstChildNode*, and *VisitOtherSiblings*. The last three sub-procedures could in turn be further decomposed, if needed.

The functional concepts allow for a high-level description of the algorithm and mask implementation details, which would be difficult to express in the ontology using relations and attributes. As an example, further decomposition of *VisitRoot* into individual steps could be stated in any of the following ways:

```
procedure VisitRoot
    DepthFirstSearch <visits> Root // using <visits> relation
end
procedure VisitFirstChildNode
    VisitRoot <visits> Root // same relation, with a more specific concept
end
```

procedure VisitOtherSiblings

Root.visited = true // that is, the value of the <visited> attribute *is modified to* true

end

As shown below, functional concepts are added to the course ontology under the root concept, *FunctionalConcept*:

```
<owl:Class rdf:about="FunctionalConcept"/>
<owl:Class rdf:about="DepthFirstSearch">
  <rdfs:subClassOf rdf:resource="FunctionalConcept"/>
</owl:Class>
<owl:Class rdf:about="VisitRoot">
  <rdfs:subClassOf rdf:resource="DepthFirstSearch"/>
</owl:Class>
<owl:Class rdf:about="VisitFirstChildNode">
  <rdfs:subClassOf rdf:resource="DepthFirstSearch"/>
</owl:Class>
<owl:Class rdf:about="VisitOtherSiblings">
  <rdfs:subClassOf rdf:resource="DepthFirstSearch"/>
</owl:Class>
```

Functional concepts are discussed in [23], in the context of courseware authoring. *Meta-functions* are also introduced, to represent relationships between primitive functions. In our case, functions are represented as functional concepts, and meta-functions, as relations. The <*is-achieved-by*> meta-function is implied in our implementation between the functional concepts and their children (instead of the explicit <*subClassOf*> relation), because it effectively indicates the steps by which the functional concepts are achieved [2]. In OeLE, the <*subClassOf*> relation is explicitly replaced with the <*is-a*> relation, and automatic assessment takes this relation into account.

---

2. Note that the domain and image of the implied <*is-achieved-by*> meta-function are reversed, because for example, DepthFirstSearch <*is-achieved-by*> VisitRoot, and not the other way around.

3.1.2 Ordering Functional Concepts

In order to assess knowledge in domains dealing with highly procedural knowledge, such as computer algorithms, it is crucial to ascertain that certain concepts are stated in a specific order. In order to address this issue, we introduce an ordering relation, which is to be applied between functional concepts. As mentioned above, relationships between functions are defined in [23] as meta-functions. The <*is-followed-by*> and <*is-preceded-by*> meta-functions define a formal specification for the temporal ordering of functions.

In [24], the *preceded_by* relation is defined similarly to <*is-preceded-by*>, and can be used to order any pair of classes P and P_1. In other words, P <*preceded_by*> P_1 is defined as: "Every P is such that there is some earlier P_1". This relation is defined as transitive, but is neither symmetric, reflexive, nor antisymmetric.

In [25], an irreflexive and transitive <*precedes*> relation is used when "the sequence of the related events is of utmost importance for the correct interpretation". This paper also defines the inverse relation, <*follows*>.

Similarly, the "Time Ontology in OWL" working draft of the World Wide Web Consortium (W3C) [26] describes a temporal ontology, OWL-Time, which defines the three following classes:

```
:Instant
      a       owl:Class ;
      rdfs:subClassOf :TemporalEntity .
:Interval
      a       owl:Class ;
      rdfs:subClassOf :TemporalEntity .
:TemporalEntity
      a       owl:Class ;
      rdfs:subClassOf :TemporalThing ;
      owl:equivalentClass
              [ a       owl:Class ;
                owl:unionOf (:Instant :Interval)
              ] .
```

A temporal ordering relation is also defined:

> There is a $before$ relation on temporal entities, which gives directionality to time. If a temporal entity T_1 is before another temporal entity T_2, then the end of T_1 is before the beginning of T_2. Thus, $before$ can be considered to be basic to instants and derived for intervals.

In our implementation, we define the ***<is-followed-by>*** meta-function in OWL as a relation (object property) having a functional concept as domain and as range:

```
<owl:ObjectProperty rdf:about="IsFollowedBy">
  <rdfs:domain rdf:resource="FunctionalConcept"/>
  <rdfs:range rdf:resource="FunctionalConcept"/>
</owl:ObjectProperty>
```

For every algorithm, a separate (meta) ontology lists the required orderings specific to that algorithm. For *DepthFirstSearch*, two instances of the relation are needed. One instance is needed between *VisitRoot* and *VisitFirstChildNode*, because the root has to be visited first, and another between *VisitFirstChildNode* and *VisitOtherSiblings*, because the first child node should be visited first. The meta-ontology is as follows:

```
VisitRoot <is-followed-by> VisitFirstChildNode
VisitFirstChildNode <is-followed-by> VisitOtherSiblings
```

Note that the following relation is also inferred by the transitive property:

```
VisitRoot <is-followed-by> VisitOtherSiblings [inferred]
```

Although there are many algorithms for graph exploration, functional concepts need only be defined once in the course ontology, and their ordering can then be declared in the meta-ontology. For instance, the *BreadthFirstSearch* algorithm could be defined with the same functional concepts as above, only with different ordering.

3.2 Natural Language Processing

All students' answers and the teacher's ideal answer are supplied in natural language texts, which need to be analyzed by the automatic grading system. The NLP stage consists of two sub-phases: *Preparation* and *Search*. The *Preparation* phase consists of spell-checking, sentence detection, tokenization, and POS tagging. In the *Search* phase, the ideal answer's concepts (a subset of the ontology's concepts) are matched to zero or more linguistic expressions in the student answer. Since 60% of the students in our experiments are non-native French speakers, we manually spell-checked the answers, making minor grammatical revisions before processing the texts.

As an example, we give an actual question from a computer algorithms course at our university: "Describe Depth-First Search". Table I-3 shows the annotation set (at the end of the NLP phase) of the teacher's ideal answer: "Depth First Search explores a graph recursively. Starting at the root, it visits the first child node before visiting its siblings." Student and ideal (teacher-provided) answers are similarly annotated; however, for every concept of the ideal answer, the teacher also supplies a numerical value indicating the relative importance of that concept within the question (this particular question would be worth $1 + 3 + 3 + 3 = 10$ points in total).

Table I-3 Example ideal answer to describe Depth-First Search

Functional concept	**Linguistic expression**	**Weight**
DepthFirstSearch	"Depth First Search"	1
VisitRoot	"Starting at the root"	3
VisitFirstChildNode	"it visits the first child node"	3
VisitOtherSiblings	"visiting its siblings"	3

3.3 Grading

The grading stage consists of calculating the semantic distance between the annotation sets (obtained in the previous phase) of each student's answer and that of the teacher's ideal answer,

with respect to the course ontology. We focus our experiments on procedural knowledge, using functional concepts and their ordering relations. The reader is advised to consult [8], or examine an easy-to-follow example in [27], for full factual and conceptual knowledge assessment in OeLE and more detailed formulas.

For each of the exam questions, the students' answers are compared to the teacher's ideal answer. For a student answer R, conceptual grades, $CG(R)$, are awarded for the ideal answer's concepts most similar to the student's concepts, in proportion to that concept's importance in the ideal answer. Each of the ideal answer's concepts can contribute at most once toward the final score.

The formulas used for calculating the final grade are given below. The formulas return values in the $[0, 1]$ interval. Teacher-provided constants allow for certain elements to be weighted according to their importance, and a severity threshold can also be specified. The best combination of these constants is problem-dependent.

3.3.1 Conceptual Grading

In order to accomplish the evaluation of conceptual knowledge, each of the concepts of the student's answer is associated with the closest concept of the ideal answer. The concept similarity, CS, between concepts c_i of the student answer and c_j of the ideal answer, is calculated using Equation I-1. The highest similarity value found (1.0 meaning *identical*) is multiplied by the relative weight of the concept in the ideal answer, and this value is then added to the conceptual grade.

$$CS(c_i, c_j) = cp \times CP(c_i, c_j) + lp \times LP(c_i, c_j) \tag{I-1}$$

The constants cp and lp are in the $[0, 1]$ interval and indicate the relative importance of the corresponding elements. Their sum should equal to 1.0. In our experiments, we set cp to 0.9 (very important), and lp to 0.1 (less important). We set the severity threshold to 0.75; if the concept similarity falls below this threshold, it is given a null value. The concept proximity, CP, is calculated using the taxonomy formed in the ontology by the class hierarchy. If concepts c_i and c_j have no taxonomic parent (other than the root), this value is zero; otherwise, it is defined as such:

$$CP(c_i, c_j) = 1 - |nodes(c_i, c_j)|/|concepts|, \tag{I-2}$$

where $|nodes(c_i, c_j)|$ is the number of concepts separating c_i and c_j through the shortest common path in the taxonomic tree, and $|concepts|$ is the total number of concepts in the ontology.

The lexical proximity, $LP(c_i, c_j)$, uses the Levenshtein distance, written $L(c_i, c_j)$ below, between the corresponding linguistic expressions of concepts c_i and c_j in the student and ideal answers, and is defined as follows:

$$LP(c_i, c_j) = 1 - (1 + L(c_i, c_j)). \tag{I-3}$$

3.3.2 Functional Grading

Because the order of the functional concepts is not factored in the conceptual grade, any permutation of the linguistic expressions of the student's answer yields the same grade. The final grade of a student answer, $FG(R)$, thus takes into account the algorithm-specific orderings of the meta-ontology. The final grade for answer R is proportional to the conceptual grading of the answer, $CG(R)$:

$$FG(R) = CG(R) \times (1 - od(1 - OF(R))), \tag{I-4}$$

where od is a constant in the $[0, 1]$ interval allowing the teacher to adjust the relative importance of the correct ordering of concepts. We set the od constant to 0.2 for all our experiments. The ordering factor of the answer, $OF(R)$, is defined as follows:

$$OF(R) = RO(R)/|orderings|, \tag{I-5}$$

where $RO(R)$ represents the number of functional concepts in the *right order* in the student answer R, and $|orderings|$ the total number of functional concepts orderings (including transitively

inferred orderings) in the meta-ontology. If no orderings are defined, $OF(R)$ is set to 1.0, and only the conceptual grades are awarded.

We define the *right order* by taking literally the definition given in [26]: if concept c_m should appear before concept c_n, then the end position (expressed here in number of tokens) of c_m should be before the beginning (also in number of tokens) of c_n.

Note that the total number of orderings defined in the meta-ontology affect the individual student grades. If there are only a few orderings, students are strongly penalized for every mistake, which is quite realistic. This is also the case with the concept proximity defined in Equation I-2, where the number of concepts in the ontology influences students' grades. However, the course and meta-ontologies are fixed during evaluation; the students' grades are therefore affected uniformly.

4 Working Example

Using Depth-First Search as an example, we apply the grading algorithm to this student answer: "Depth First Search explores a graph recursively. It visits the first child node **after** visiting its siblings." Table I-4 shows the produced functional concepts. The use of the word "after" means that the last two concepts' positions should be inverted, yielding the answer shown in Table I-5.

After inverting the last two concepts' positions, this ordering is invalidated:

```
VisitFirstChildNode <is-followed-by> VisitOtherSiblings
```

Also, the student failed to mention the *VisitRoot* functional concept. For this reason, these two orderings are invalid, because their domain is missing:

```
VisitRoot <is-followed-by> VisitFirstChildNode
VisitRoot <is-followed-by> VisitOtherSiblings [inferred]
```

The conceptual grading of this answer is then: $(1 + 0 + 3 + 3)/10 = 70\%$. Setting the ordering constant, od, to 0.5, the final grade becomes:

$$FG(R) = 70\% \times (1 - 0.5(1 - 0)) = 35\%, \tag{I-6}$$

Table I-4 Example annotated answer of a student to describe Depth-First Search

Functional concept	**Linguistic expression**	**Position (token #)**
DepthFirstSearch	"Depth First Search"	1–3
VisitRoot	—	—
VisitFirstChildNode	"it visits the first child node"	8–13
VisitOtherSiblings	"visiting its siblings"	15–17

Table I-5 Annotated answer of a student to describe Depth-First Search, after inversion

Functional concept	**Linguistic expression**	**Position (token #)**
DepthFirstSearch	"Depth First Search"	1–3
VisitRoot	—	—
VisitFirstChildNode	"it visits the first child node"	**15–17**
VisitOtherSiblings	"visiting its siblings"	**8–13**

Inverted positions are highlighted in bold.

where the conceptual grading is 70%, and the ordering factor is zero. Given that this algorithm contains only three functional concept orderings (one is inferred), and that none is correct, this low grade seems acceptable, or at least a reasonable improvement over the former grade of 70% that would have been attributed by a strictly factual conceptual grading system.

5 Experimental Results

To evaluate our system, we used answers from three exams on computer algorithms given at Université de Moncton between 2011 and 2013. The questions respectively asked to detail the operation of Dijkstra's shortest path algorithm ($\mathcal{DIJ}$), depth-first search ($\mathcal{DFS}$), and B+ tree search ($\mathcal{BT}$). Each question was answered respectively by 13, 22, and 18 students, with an average length of 91, 40, and 82 words [3]. Our system's grading results were compared to grades supplied by 3 instructors. The average, standard deviation, and Pearson correlation were used for this comparison. Regression analysis was performed on the data to determine the statistical significance.

Our system uses two dictionaries. The first one is a thesaurus containing 244 synonyms. The second one contains 73 (for $\mathcal{DIJ}$), 34 (for $\mathcal{DFS}$), and 136 (for $\mathcal{BT}$) linguistic expressions associated to course ontology concepts that were discovered during annotation. The system recognizes 262 verbs, 370 nouns, 155 adjectives, 141 adverbs, and 48 other word categories, such as articles, prepositions, etc. The course ontology contains 22 concepts and 1 relation (*is-achieved-by*), shared by the three questions (see Figure I-1).

We also compare our results to a version of OeLE that we implemented according to [8, 27]. OeLE's grading algorithm normally takes into account the similarity of concepts, relations, and attributes. Since our course ontology does not contain attributes, and because the *is-achieved-by* relation is only used to indicate functional concept decomposition, relations and attributes are not assessed. The relative weight of the concepts, given in Tables I-6, I-7, and I-8, were set by the human graders before running the experiments. In the remainder of this section, we simply refer to our version of OeLE as "OeLE", and to our system as "System".

3. We do not disclose the students' answers here, for confidentiality reasons. The students' identity was also not revealed to the human graders.

Figure I-1 Course ontology used for automatic assessment

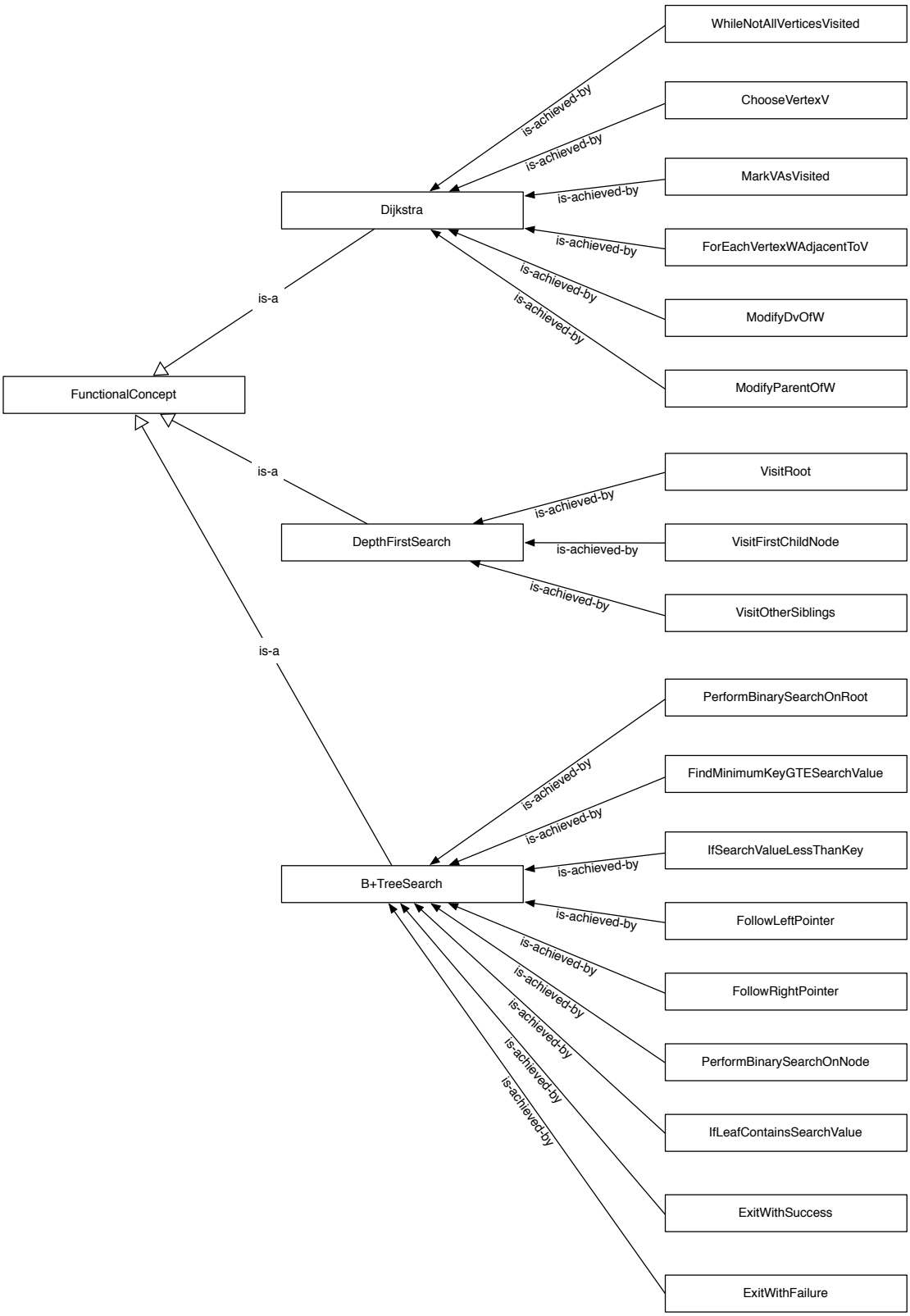


Table I-6 Relative weights of concepts for $\mathcal{DFS}$

Concept	Relative weight
DepthFirstSearch	10%
VisitRoot	30%
VisitFirstChildNode	30%
VisitOtherSiblings	30%

Table I-7 Relative weights of concepts for $\mathcal{DIJ}$

Concept	Relative weight
WhileNotAllVerticesVisited	20%
ChooseVertexV	20%
MarkVAsVisited	10%
ForEachVertexWAdjacentToV	20%
ModifyDvOfW	20%
ModifyParentOfW	10%

Table I-8 Relative weights of concepts for $\mathcal{BT}$

Concept	Relative weight
PerformBinarySearchOnRoot	20%
FindMinimumKeyGTESearchValue	20%
IfSearchValueLessThanKey	10%
FollowLeftPointer	5%
FollowRightPointer	5%
PerformBinarySearchOnNode	20%
IfLeafContainsSearchValue	10%
ExitWithSuccess	5%
ExitWithFailure	5%

For $\mathcal{DIJ}$, Table I-9 shows the grades, over 4 points, given by human graders (G1–G3) for each student (S1–S13). The instructors gave average grades between 3.00 and 3.31, with an average of 3.15 and a standard deviation of 0.80.

Table I-9 Grades given by human graders for $\mathcal{DIJ}$

Student	**G1**	**G2**	**G3**	**Avg-Human**	**Stdev**
S1	3	2.8	2	2.60	0.53
S2	4	3.6	4	3.87	0.23
S3	4	3.6	4	3.87	0.23
S4	3	3.6	3	3.20	0.35
S5	4	3.6	3.5	3.70	0.26
S6	3	2.4	2.5	2.63	0.32
S7	4	4	4	4.00	0.00
S8	3.5	3.2	3	3.23	0.25
S9	1	2	1	1.33	0.58
S10	4	3.2	3	3.40	0.53
S11	4	3.6	2.5	3.37	0.78
S12	1.5	2	2.5	2.00	0.50
S13	4	3.2	4	3.73	0.46
Avg	3.31	3.14	3.00	3.15	—
Stdev	1.01	0.65	0.91	0.80	—

Table I-10 shows a comparison of the average human grades for each student with those of OeLE and those of our system for $\mathcal{DIJ}$. The system gave an average of 2.83 points (out of 4), with a standard deviation of 0.80. The correlation between the average results of human graders and those of our system is 0.59 (with $p < 0.05$), which indicates a positive relationship. OeLE obtains a lower correlation of 0.52 with the average results of human graders, although it gives average grades (3.10) which are closer to those of human graders (3.15) [4]. The difference between

4. Recall that Equation I-4 penalizes students for concepts in the wrong order, which explains why the average grade of our system is systematically lower.

Avg-Human and OeLE's grades is given in the Diff-OeLE column, while the difference between Avg-Human and our system is given in the Diff-System column.

Table I-10 Comparison of average human grades with those of OeLE and our system for $\mathcal{DIJ}$

Student	**Avg-Human**	**OeLE**	**Diff-OeLE**	**System**	**Diff-System**
S1	2.60	1.08	−1.52	0.91	−1.69
S2	3.87	3.60	−0.27	3.60	−0.27
S3	3.87	3.60	−0.27	3.60	−0.27
S4	3.20	3.60	0.40	3.39	0.19
S5	3.70	3.60	−0.10	3.34	−0.36
S6	2.63	2.52	−0.11	2.23	−0.40
S7	4.00	3.60	−0.40	3.34	−0.66
S8	3.23	3.60	0.37	2.93	−0.30
S9	1.33	3.24	1.91	2.82	1.49
S10	3.40	3.60	0.20	3.19	−0.21
S11	3.37	2.52	−0.85	2.23	−1.13
S12	2.00	2.16	0.16	1.91	−0.09
S13	3.73	3.60	−0.13	3.34	−0.39
Avg	3.15	3.10		2.83	
Stdev	0.80	0.80		0.80	

Figure I-2 compares the grades given by human graders to those of OeLE and our system for $\mathcal{DIJ}$. Vertical bars indicate the minimum and maximum grades given to this student by the human graders. Figure I-3 shows the correlation of the average human grades with those given by the system. The diagonal ($y = x$) indicates perfect agreement.

Figure I-2 Comparison of human grades with those of OeLE and our system for $\mathcal{DIJ}$

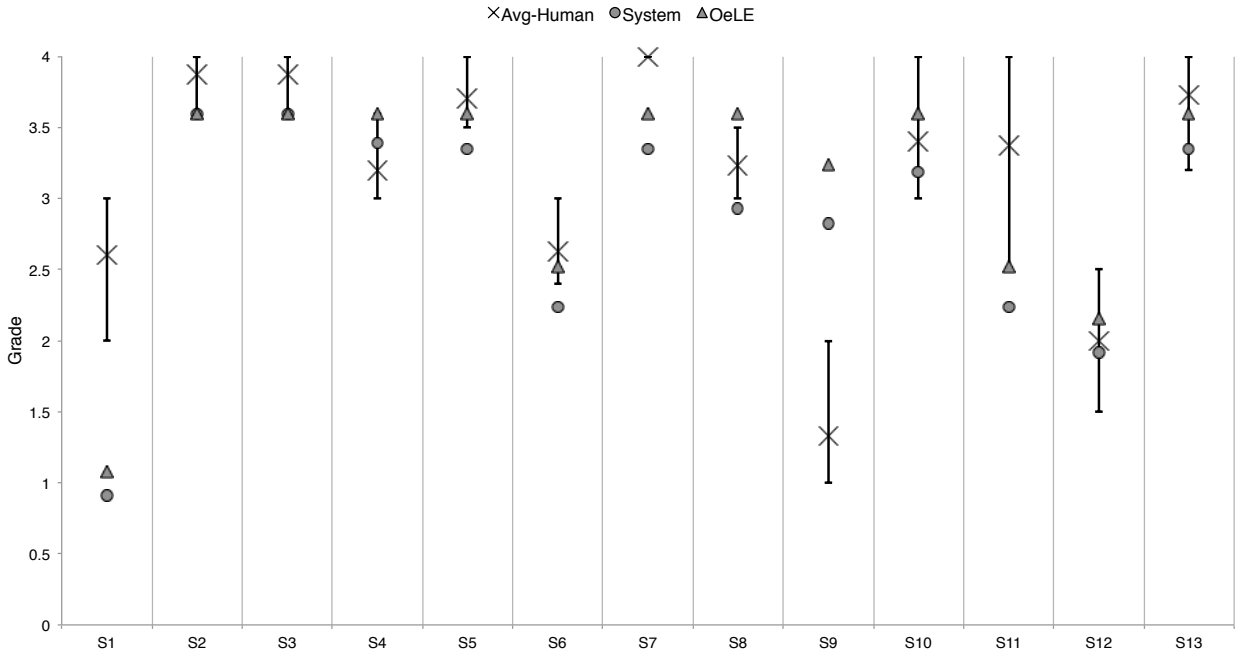


Figure I-3 Correlation of human grades with those of the system for $\mathcal{DIJ}$

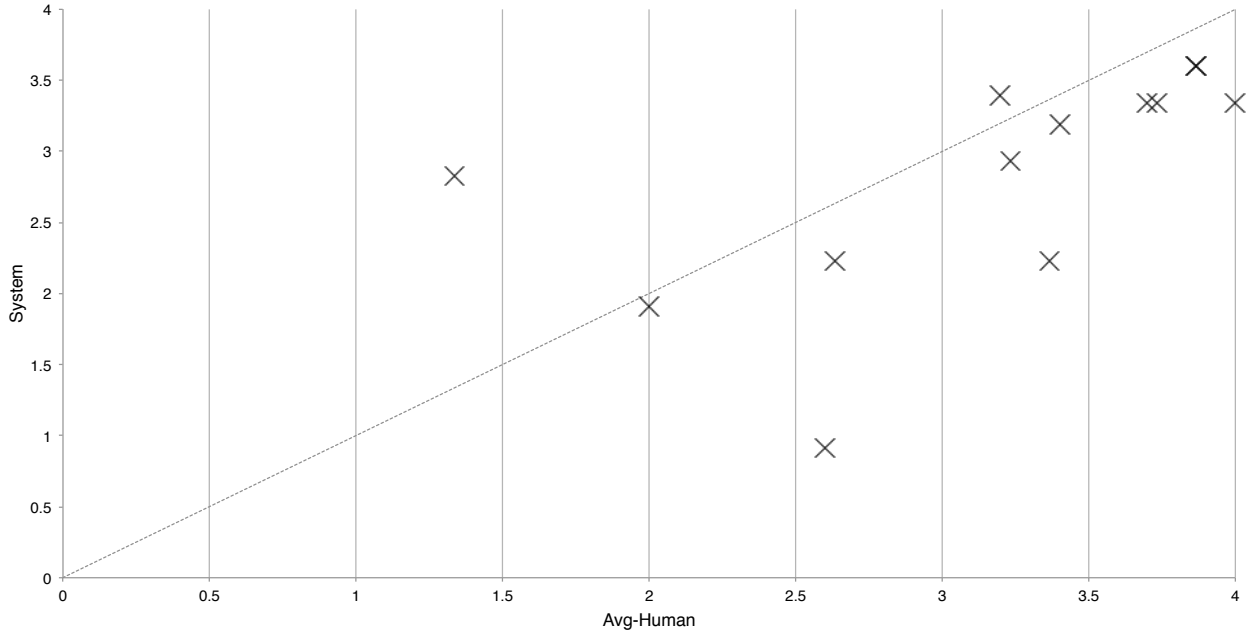


Table I-11 shows the ordering factor, $OF(R)$, for each student answer R, as well as the average grades given by humans graders. See Annex A for the list of expected orderings for each algorithm. For $\mathcal{DIJ}$, the correlation between the average human grades and the ordering factor is 0.56, which is in fact better than the correlation between OeLE's grades and those of our system (0.52).

Table I-11 Ordering factor of student answers and average human grades for $\mathcal{DIJ}$

Student	**Avg-Human**	**Ordering factor**
S1	2.60	0.21
S2	3.87	1.00
S3	3.87	1.00
S4	3.20	0.71
S5	3.70	0.64
S6	2.63	0.43
S7	4.00	0.64
S8	3.23	0.07
S9	1.33	0.36
S10	3.40	0.43
S11	3.37	0.43
S12	2.00	0.43
S13	3.73	0.64

For $\mathcal{DFS}$, Table I-12 shows the grades, over 5 points, given by human graders (G1–G3) to the 22 students who answered. The instructors gave average grades between 2.36 and 3.07, with an average of 2.69 and a standard deviation of 1.50.

Table I-13 shows a comparison of the average human grades with those of OeLE and those of our system for $\mathcal{DFS}$. Our system gave an average of 1.59 points (out of 5), with a standard deviation of 1.35. The correlation between the system and the average human grades is 0.67 ($p < 0.05$), which indicates a positive correlation. In this experiment, OeLE's correlation with the average human grades is slightly higher, at 0.69.

Table I-12 Grades given by human graders for $\mathcal{DFS}$

Student	**G1**	**G2**	**G3**	**Avg-Human**	**Stdev**
S1	5	5	5	5.00	0.00
S2	3	2	1	2.00	1.00
S3	0	0	0	0.00	0.00
S4	3.5	4	4	3.83	0.29
S5	3	3.5	3	3.17	0.29
S6	2.5	5	1	2.83	2.02
S7	1	2	1	1.33	0.58
S8	5	5	4	4.67	0.58
S9	4.5	4	3	3.83	0.76
S10	2	3.5	2	2.50	0.87
S11	1	1.5	2	1.50	0.50
S12	1	0	0	0.33	0.58
S13	2.5	3.5	2	2.67	0.76
S14	0	0	0	0.00	0.00
S15	1	2	2	1.67	0.58
S16	4.5	5	4	4.50	0.50
S17	2.5	4	3.5	3.33	0.76
S18	4.5	4	3.5	4.00	0.50
S19	4	4	3	3.67	0.58
S20	4.5	4.5	3	4.00	0.87
S21	1	1	2	1.33	0.58
S22	2	4	3	3.00	1.00
Avg	2.64	3.07	2.36	2.69	—
Stdev	1.63	1.71	1.57	1.50	—

Table I-13 Comparison of average human grades with those of OeLE and our system for $\mathcal{DFS}$

Student	**Avg-Human**	**OeLE**	**Diff-OeLE**	**System**	**Diff-System**
S1	5.00	3.15	−1.85	2.73	−2.27
S2	2.00	1.80	−0.20	1.44	−0.56
S3	0.00	0.45	0.45	0.36	0.36
S4	3.83	4.50	0.67	4.50	0.67
S5	3.17	0.45	−2.72	0.36	−2.81
S6	2.83	0.45	−2.38	0.36	−2.47
S7	1.33	1.80	0.47	1.44	0.11
S8	4.67	3.15	−1.52	2.73	−1.94
S9	3.83	4.50	0.67	4.50	0.67
S10	2.50	1.80	−0.70	1.44	−1.06
S11	1.50	1.80	0.30	1.44	−0.06
S12	0.33	0.00	−0.33	0.00	−0.33
S13	2.67	0.45	−2.22	0.36	−2.31
S14	0.00	0.45	0.45	0.36	0.36
S15	1.67	0.45	−1.22	0.36	−1.31
S16	4.50	1.80	−2.70	1.44	−3.06
S17	3.33	2.70	−0.63	2.16	−1.17
S18	4.00	3.15	−0.85	2.73	−1.27
S19	3.67	3.15	−0.52	2.73	−0.94
S20	4.00	3.15	−0.85	2.73	−1.27
S21	1.33	0.45	−0.88	0.36	−0.97
S22	3.00	0.45	−2.55	0.36	−2.64
Avg	2.69	1.82		1.59	
Stdev	1.50	1.42		1.35	

Figure I-4 compares the grades given by human graders to those of OeLE and our system for $\mathcal{DFS}$. Figure I-5 shows the correlation of the average human grades with those given by the system.

Figure I-4 Comparison of human grades with those of OeLE and our system for $\mathcal{DFS}$

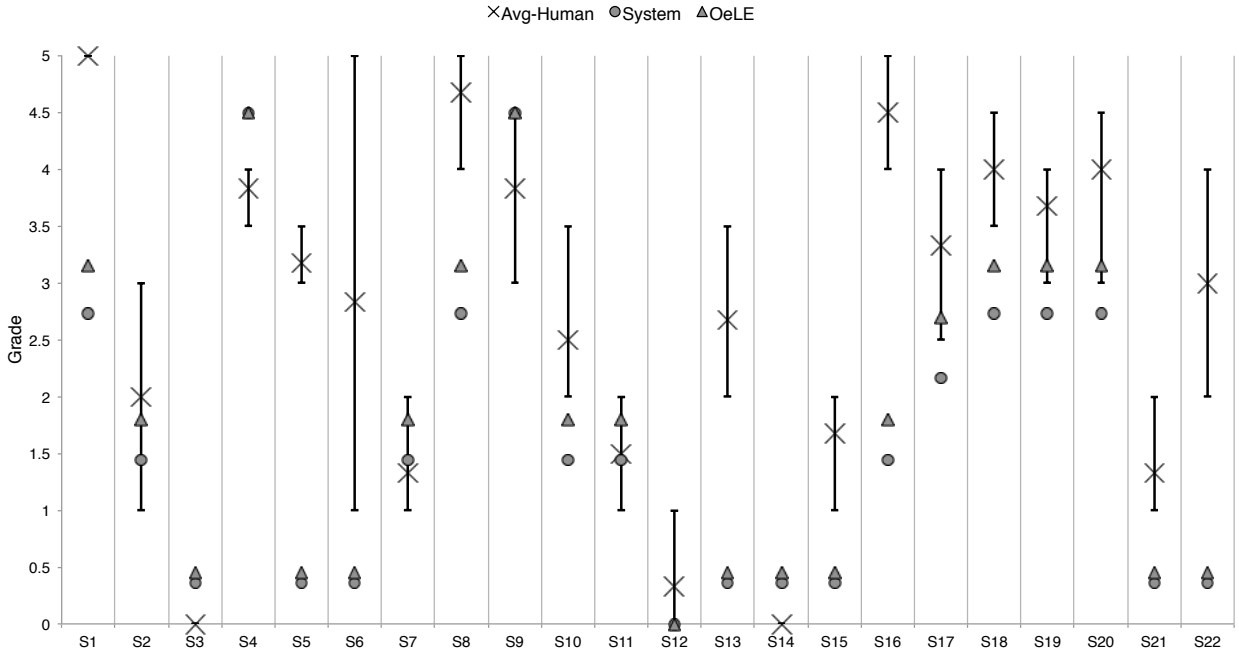


Table I-14 shows the ordering factor, $OF(R)$, for each student answer R, as well as the average grades given by humans graders. See Annex A for the list of expected orderings for each algorithm. For $\mathcal{DFS}$, the correlation between the average human grades and the ordering factor is 0.51, which is lower than the correlation between OeLE's grades and those of our system (0.69). It seems that the human graders were giving more importance to the presence of the (ideal answer's) concepts in the texts, than to the order in which they appeared.

For BT, Table I-15 shows the grades, over 10 points, given by human graders (G1–G3) for each student (S1–S18). The instructors gave average grades between 5.28 and 6.25, with an average of 5.84 and a standard deviation of 2.26.

Table I-16 shows a comparison of the average human grades with those of OeLE and those of our system for $\mathcal{BT}$. Our system gave an average of 5.67 points (out of 10), with a standard deviation of 1.92. The correlation between the system and the average human grades is 0.86 ($p < 0.05$),

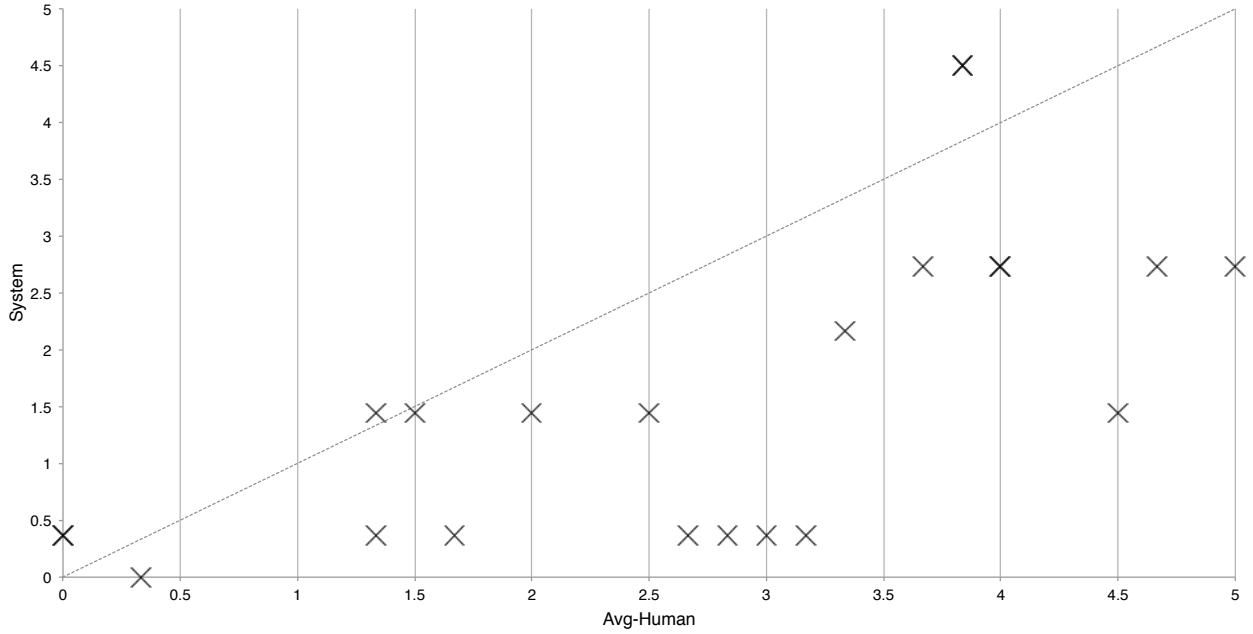
Table I-14 Ordering factor of student answers and average human grades for $\mathcal{DFS}$

Student	Avg-Human	Ordering factor
S1	5.00	0.33
S2	2.00	0.00
S3	0.00	0.00
S4	3.83	1.00
S5	3.17	0.00
S6	2.83	0.00
S7	1.33	0.00
S8	4.67	0.33
S9	3.83	1.00
S10	2.50	0.00
S11	1.50	0.00
S12	0.33	0.00
S13	2.67	0.00
S14	0.00	0.00
S15	1.67	0.00
S16	4.50	0.00
S17	3.33	0.00
S18	4.00	0.33
S19	3.67	0.33
S20	4.00	0.33
S21	1.33	0.00
S22	3.00	0.00

Table I-15 Grades given by human graders for $\mathcal{BT}$

Student	**G1**	**G2**	**G3**	**Avg-Human**	**Stdev**
S1	8.5	7	8	7.83	0.76
S2	9	7.5	8.5	8.33	0.76
S3	5	5	6	5.33	0.58
S4	8	5	6	6.33	1.53
S5	6	8	8.5	7.50	1.32
S6	10	7	8.5	8.50	1.50
S7	2.5	2	2	2.17	0.29
S8	9	7.5	7	7.83	1.04
S9	10	9	10	9.67	0.58
S10	5.5	4	7	5.50	1.50
S11	5	7	5	5.67	1.15
S12	3	2	3	2.67	0.58
S13	3.5	4	3	3.50	0.50
S14	4	2	5	3.67	1.53
S15	7	7	8.5	7.50	0.87
S16	5.5	5	7.5	6.00	1.32
S17	3	5	5	4.33	1.15
S18	3.5	1	4	2.83	1.61
Avg	6.00	5.28	6.25	5.84	—
Stdev	2.55	2.39	2.29	2.26	—

Figure I-5 Correlation of human grades with those of the system for $\mathcal{DFS}$



which indicates a positive correlation. In this experiment, OeLE's correlation with the average human grades is the same.

Figure I-6 compares the grades given by human graders to those of OeLE and our system for $\mathcal{BT}$. Figure I-7 shows the correlation of the average human grades with those given by the system.

Table I-17 shows the ordering factor, $OF(R)$, for each student answer R, as well as the average grades given by humans graders. See Annex A for the list of expected orderings for each algorithm. For $\mathcal{BT}$, the correlation between the average human grades and the ordering factor is 0.73, which is lower than the correlation between OeLE's grades and those of our system (0.86). We can interpret this result in two ways. The first explanation is that the order in which the concepts appear in the texts is not *crucial* to these human graders. The other explanation is that our measure is not completely accurate. However, by looking at the highest value (1.00, for student S6), we notice that the first grader gave a perfect score (see Table I-15), and that the concepts were in fact in the expected order. The lowest values (zero, for students S7 and S12), also seem accurate. Student S7 explained that the search algorithm starts at the leaves, and goes up the tree to the root, which is

Table I-16 Comparison of average human grades with those of OeLE and our system for $\mathcal{BT}$

Student	**Avg-Human**	**OeLE**	**Diff-OeLE**	**System**	**Diff-System**
S1	7.83	7.20	– 0.63	6.79	– 1.04
S2	8.33	8.55	0.22	7.82	– 0.52
S3	5.33	4.95	– 0.38	4.10	– 1.23
S4	6.33	7.20	0.87	6.58	0.25
S5	7.50	6.30	– 1.20	5.58	– 1.92
S6	8.50	9.00	0.50	9.00	0.50
S7	2.17	4.50	2.33	3.60	1.43
S8	7.83	8.55	0.72	7.94	0.11
S9	9.67	8.10	– 1.57	7.41	– 2.26
S10	5.50	5.40	– 0.10	5.01	– 0.49
S11	5.67	7.20	1.53	6.48	0.81
S12	2.67	2.70	0.03	2.16	– 0.51
S13	3.50	3.15	– 0.35	2.66	– 0.85
S14	3.67	4.50	0.83	3.86	0.19
S15	7.50	7.20	– 0.30	6.69	– 0.81
S16	6.00	7.20	1.20	6.69	0.69
S17	4.33	4.95	0.62	4.31	– 0.02
S18	2.83	5.85	3.02	5.35	2.52
Avg	5.84	6.25		5.67	
Stdev	2.26	1.86		1.92	

Figure I-6 Comparison of human grades with those of OeLE and our system for $\mathcal{BT}$

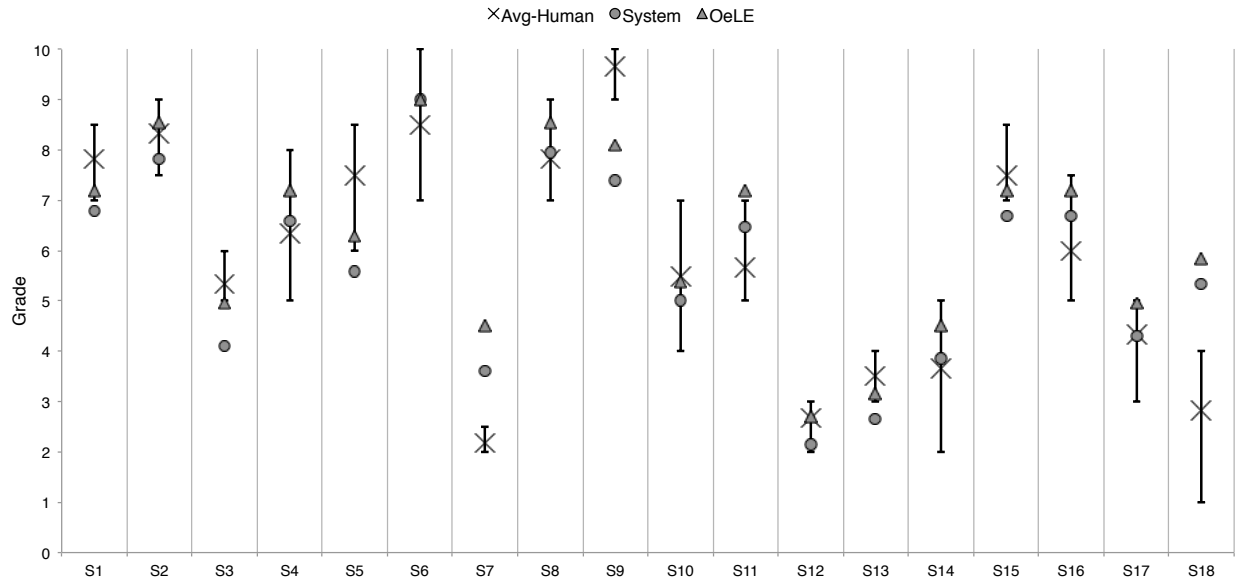
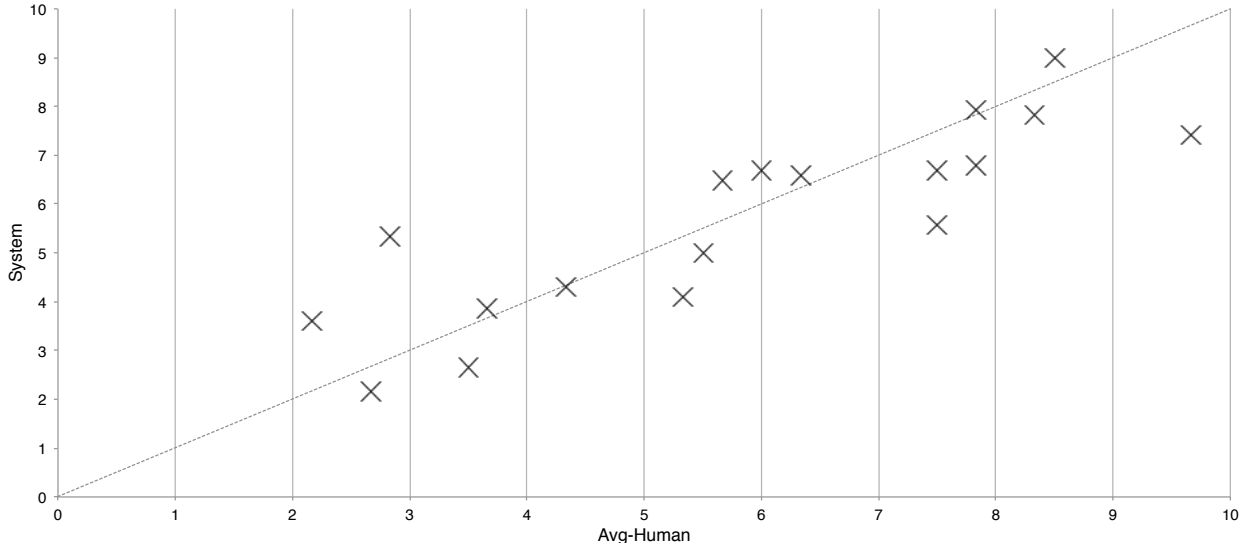


Figure I-7 Correlation of human grades with those of the system for $\mathcal{BT}$



false. Student S12 gave a very short answer (28 words), which incorrectly stated that the search is performed only on the internal nodes. Both students received low average grades (2.17 and 2.67 out of 10 points, respectively) from human graders.

Table I-17 Ordering factor of student answers and average human grades for $\mathcal{BT}$

Student	**Avg-Human**	**Ordering factor**
S1	7.83	0.71
S2	8.33	0.57
S3	5.33	0.14
S4	6.33	0.57
S5	7.50	0.43
S6	8.50	1.00
S7	2.17	0.00
S8	7.83	0.64
S9	9.67	0.57
S10	5.50	0.64
S11	5.67	0.50
S12	2.67	0.00
S13	3.50	0.21
S14	3.67	0.29
S15	7.50	0.64
S16	6.00	0.64
S17	4.33	0.36
S18	2.83	0.57

Overall, the system gives grades with a positive correlation to human grades. We can observe that the system performs well on answers of variable length and different granularity levels.

6 Conclusion

The work presented in this paper brings the novelty of procedural knowledge assessment in free-text answers. The experimental evaluation was carried on three exam questions about computer algorithms given at Université de Moncton between 2011 and 2013. The questions were answered by a total of 53 students and the answers were graded by 3 instructors. Experimental results show correlations of 0.59, 0.67, and 0.86 between the system's scores and those of human graders.

The approach put forth in this paper introduces functional concepts to represent procedural knowledge in ontologies. The class hierarchy of functional concepts is considered as a series of instances of the <*is-achieved-by*> relation. For every computer algorithm (or procedure, in other domains), a series of instances of the <*is-followed-by*> relation specify an ordering for pairs of functional concepts. This approach could also be used in other domains where procedural knowledge is central to processing the text.

Future work could make use of other relations (meta-functions) defined in [23]. The <*requires*>, <*is-a-prerequisite-for*>, <*follows-from*>, and <*is-assigned-to*> meta-functions could allow for a more elaborate description of computer algorithms. For example, steps joined with the <*requires*> relation would be more strongly penalized when missing.

The <*if-[goal]-then-[action]*> meta-function could also be considered for flow control structures, such as loops or branches. However, automatic assessment of such structures in free-text answers, which lack proper indentation or braces, could prove difficult.

Acknowledgments

We thank the Canada Foundation for Innovation for financing the infrastructure and the technical support for this work.

This work was conducted using the Protégé resource, which is supported by grant GM10331601 from the National Institute of General Medical Sciences of the United States National Institutes of Health.

CHAPITRE II

ASSESSING PROCEDURAL KNOWLEDGE IN FREE-TEXT ANSWERS THROUGH A HYBRID SEMANTIC WEB APPROACH

ÉVALUATION DES CONNAISSANCES PROCÉDURALES DANS LES RÉPONSES EN TEXTE LIBRE PAR UNE APPROCHE HYBRIDE BASÉE SUR LE WEB SÉMANTIQUE

Abstract

Several techniques have been proposed to automatically grade students' free-text answers in e-learning systems. However, these techniques provide no or limited support for the evaluation of acquired procedural knowledge. To address this issue, we propose a new approach, named PROC-MARK, specifically designed to assess answers containing procedural knowledge. It requires a teacher to provide the ideal answer as a semantic network that is used to automatically score learners' answers in plain text. The novelty of our approach resides mainly in three areas: a) the variable granularity levels possible in the semantic network and the parameterizing of ontology concepts, thus allowing the students free expression of their ideas; b) the new similarity measures of the grading system that give refined numerical scores; c) the language-independence of the grading system as all linguistic information is given as data files or dictionaries and is distinct of the semantic knowledge of the semantic network. Experimental results in a Computer Algorithms course show that the approach gives marks that are very close to those of human graders, with positive correlations (0.70, 0.79, and 0.79).

Keywords: procedural knowledge; computer-assisted assessment; free-text answers; ontology; semantic networks; similarity measures

1 Introduction

Assessing the students' learning in an e-learning environment often relies on multiple choice or fill-in-the-blank questions, which only trigger the lowest level (*Knowledge*) of Bloom's taxonomy of knowledge acquisition [7]. Several attempts have been made to incorporate open-ended questions in online assessment, which would favour triggering the higher levels of Bloom's taxonomy (*Synthesis* and *Evaluation*) in the students' learning. A variety of techniques have been used to develop e-learning environments that can automatically grade free-text answers, such as Natural Language Processing (NLP) [14, 18, 28], statistical techniques [20–22], or Information Extraction (IE) [11,12]. Although human intervention is still required in the grading process, automatic assessment is much faster and is done more objectively than manual scoring. Techniques for automatic assessment perform well on texts containing factual and conceptual knowledge (that is, description

of events, places, etc.). However, they suffer from an important limitation: they generally provide no or limited support for the evaluation of procedural knowledge (e.g. [8,19]). For several domains, it is not only crucial to evaluate the facts stated in a text, but also the temporal relationships between the steps of the procedure expressed through the statements. If these relationships are ignored, texts containing the expected keywords (even in the wrong order) are still awarded high marks. For example, in Computer Science, to exchange the contents of two variables a and b, the three steps: $c := a$, $a := b$, and $b := c$ have to be stated in the student answer in this order. Procedural knowledge is very common in many domains, such as cooking, call center troubleshooting and mechanical repairs.

To address this issue, we propose a novel approach, named PROCMARK, specifically designed for the assessment of free-text answers containing procedural knowledge. The system requires a teacher to provide an ideal answer for each question as a semantic network, based on a course ontology. The expected student answer is a free text, around a paragraph in length, explaining the operation of a computer algorithm they learned. The system utilizes the semantic network to grade answers automatically. Our system bears some similarity to OeLE's [8] grading algorithm, with the addition of three novel similarity measures that allow the assessment of procedural relationships between steps described in the texts. The semantic network representation of ideal answers is akin to MultiNet Working Bench (MWB) [19], with our system having more parameterized concepts. However, these latter systems are not specifically designed to assess procedural knowledge.

The contributions of our system are multifaceted. First, to allow the representation of procedural knowledge, we introduced in [1, 2] the notion of *functional concepts* that operate like computer programming functions or procedures. These functional concepts can be nested (decomposed into more specific concepts) and reused (multiple function calls). In this new version, we define an expanded ontology offering general algorithmic knowledge, such as flow control structures, conditions, etc., and use semantic networks to represent answers.

Second, to express and handle different granularity levels of knowledge in the ideal answer's semantic network, we chose that all knowledge representation be parameterized as concepts, namely functional concepts, flow control structures, conditions, relations, attributes, and literals. These de-

sign decisions also give students more liberty by allowing them to express their answers in more or less detail.

Third, to support these levels of granularity in students' answers, we propose an improved grading algorithm. It includes three novel similarity measures: *part-whole similarity*, which considers the nesting and composition of ontology concepts; *textual proximity*, which takes into account the distance between a concept and its related concepts as expressed in the students' answers; and *temporal ordering*, which considers the relative order between procedure steps.

Finally, to evaluate our approach, we applied it to questions in a course on computer algorithms. Experimental results show that the marks given by our system are very close to those given by human graders, with positive correlations (0.70, 0.79, and 0.79).

The remaining of this paper is organized as follows. Section 2 surveys automatic free-text assessment systems. Section 3 presents our approach, followed by our results in Section 4. We conclude the paper in Section 5 with a brief discussion and our final remarks.

2 Literature Review

This section surveys previous and ongoing research in automatic short-answer assessment. Other systems have also been developed for longer essay scoring, but will not be examined here. A detailed review can be found in [10].

Some systems require teachers to provide training sets of marked student answers. For example, Auto-marking [28] uses pattern-matching to compare students' answers to a training set of marked student answers. Comparative experiments were conducted using Inductive Logic Programming (ILP), decision tree learning and Naive Bayesian learning. This system obtained 88% of exact agreement with human grading in factual questions about a Biology course. Bayesian Essay Test Scoring System (BETSY) [14] uses Naive Bayes classifiers to search for specific features in students' answers. In a Biology course, it achieved up to 80% accuracy. In [18], a combination of NLP and Support Vector Machines (SVM) is used to classify answers into two classes (above/below 6 points out of 10). It obtained an average of 65% precision rate (the only reported metric).

Other systems require a set of reference texts. For example, Research Methods Tutor [20] uses Latent Semantic Analysis (LSA) to compare the students' answers to a set of expected answers. If the student answers incorrectly, the system guides the student into obtaining the right answer. The Willow system [21] requires several unmarked reference answers for each question. It also uses LSA to evaluate students' answers written in English or Spanish. In a Computer Science course, it achieved on average 0.54 correlation with the teacher's grading. A system currently in use at the University of Witwatersrand [22] uses LSA and clustering techniques. It achieves between 0.80 and 0.95 correlation with the teacher's grading. Their system will not process correctly "The *left* most node of the *right* subtree" and "The *right* most node of the *left* subtree". Our system recognizes these differences as it assigns different attributes to the respective nodes and subtrees in the semantic network.

Many of the previous systems use machine learning techniques, which explains the large number of examples for the training set. Some systems compare students' answers to an ideal answer supplied by the teacher. For instance, Automated Text Marker [11] uses pattern-matching. It has been tested in courses on Prolog programming, psychology and biology. Although Prolog programming is one of the domain applications, the knowledge assessed is factual rather than procedural. Automark [12] uses Information Extraction techniques to grade students' answers by comparing them to a mark scheme template provided by the teacher. It achieved 94.7% agreement with human grading for six of the seven science-related factual questions asked on a test exam.

The OeLE [8] system uses a combination of NLP techniques and Semantic Web ontologies to assess the level of understanding of the students. The teacher-provided ideal answer is annotated against the course ontology and compared to the (similarly annotated) students' answers during the assessment. The researchers expect a deeper understanding of the text than statistical techniques. The OeLE system has been used in two online courses.

The MultiNet Working Bench [19] system also attempts a deeper understanding of the text with the help of semantic networks. A separate graphical tool is used to represent the students' answers visually. It compares the semantic network extracted from the student answer to a reference semantic network submitted by the teacher. Verified parts of the network are displayed in green,

while wrong or unverified parts (not supported by logic inference) are displayed in red. MWB has been applied to the evaluation of Java, Prolog, Scheme and C programming by processing *program code* input by the student.

Pathfinder Network analysis is used in [29] to score concept maps hand-sketched by students to illustrate relationships between terms or concepts. Spatial position of terms is taken into account when automatically assessing students' knowledge. Pathfinder measures the pair-wise distance between concepts in pixels (from a rendering of the sketch in the ALA-Mapper software). We define *textual proximity* as a function of the distance between words or expressions of the students' concepts in their texts.

A hierarchical similarity measure is introduced in [30] to calculate the distance between nodes in a directed acyclic graph (DAG). The measure is used for text categorization, where the categories form a DAG. This measure is more refined than the *ontological proximity* measure used in our approach and in [8]. However, our approach relies as well on a recursive *part-whole similarity* measure to assess procedural knowledge more precisely.

Semantic similarity measures are also used in [31], a move away from bag-of-words (BOW) approaches. This approach tries to align dependency graphs, which represent relationships between concepts, in student and teacher answers, in order to calculate the grade. It was applied in a Data Structures course at the University of North Texas.

To our knowledge, no system is specifically designed to assess procedural knowledge in free-text answers, although such a system would be helpful in an e-learning environment for scientific and technical domains, where procedural knowledge is very common.

3 The PROCMARK System

In this section, we present our system, which is designed to assess free-text student answers containing procedural knowledge. The system works in four phases, which are explained in detail in the next subsections. In the first phase, a teacher builds a semantic network for each question's ideal answer by using the course ontology. In the second phase, the system annotates a student's

answer by using NLP techniques. It matches the concepts occurring in the ideal answer to words or expressions in the student's answer. The third phase links the concepts found in the annotated answer to nodes of the ideal answer's semantic network, thus forming the specific semantic network of the student's answer as an overlay. Finally, during the semantic network traversal, the system calculates the similarity measures that are used by the grading algorithm to give a numerical mark.

3.1 Representing Domain Knowledge

The first phase consists in defining the course ontology to represent the procedural knowledge and then instantiating part of it for each question as a semantic network of the ideal answer.

3.1.1 Course Ontology

The course ontology, in a Computer Algorithms course, contains the usual control structures, operators, conditions, etc. Our ontology contains 131 entries, organized as a hierarchy [1]. It is encoded in the Web Ontology Language (OWL), which allows the use of graphical editors such as Protégé, and facilitates its reuse and sharing with other teachers. Figure II-1 shows a partial concept class hierarchy. The course ontology includes 11 categories of concepts (see Table II-1) that are used for all exam questions. It is possible to define new concepts and categories, as needed, when more questions are added. A more detailed explanation of these concepts and their categories is given in the next section.

3.1.2 Semantic Network of Ideal Answer

The ideal answer for each question is given by the teacher in the form of a semantic network. The semantic network is also encoded in OWL and it imports the course ontology. The semantic network's nodes are instances of concepts defined in the course ontology. This network is a directed connected graph. Its root is always a functional concept representing a computer algorithm.

---

1. See Annex B for the full course ontology.

Figure II-1 Partial concept class hierarchy

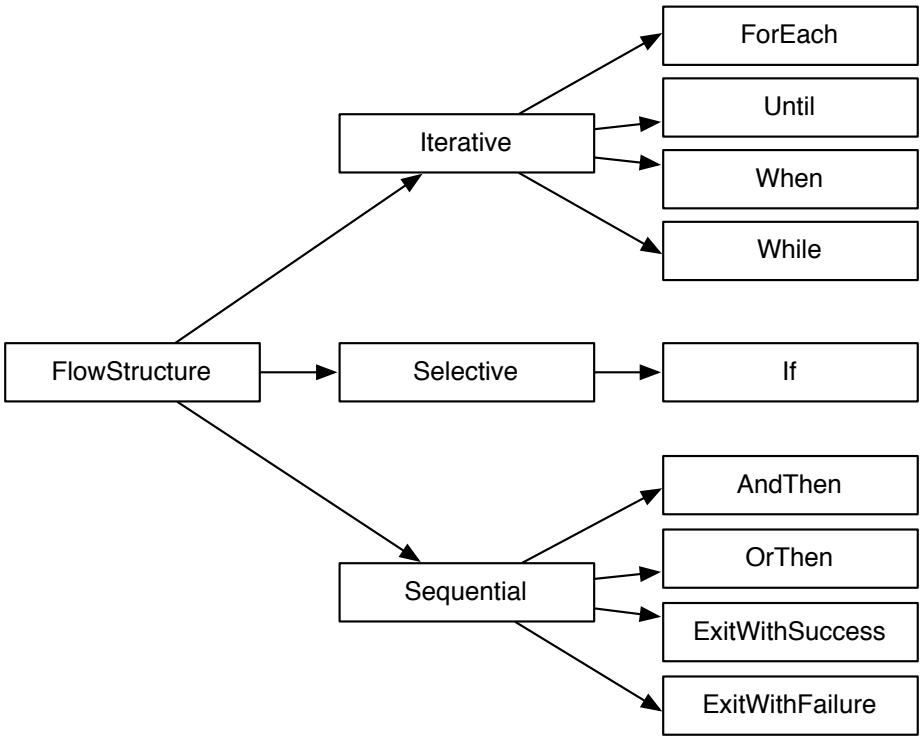


Table II-1 Categories of concepts and examples

Concept categories	**Example concepts**
FlowStructure	
Iterative	*ForEach*, *While*
Sequential	*AndThen*, *OrThen*
Selective	*If*
Operator	
Arithmetic	*Addition*, *Subtraction*
Logical	*AND*, *OR*, *NOT*
Comparison	*EqualTo*, *GreaterThan*
Member	*Dereference*
Field	Visited, Dv, Parent
Element	Vertex, Edge
Set	Vertices, Edges
Quantifier	*ForAll*, *ThereExists*
Condition	AllVerticesAreVisited, VIsVisited
Variable	V, W
FunctionalConcept	Dijkstra, B+Search, ChooseVertexV
Literal	True, False
Action	ToTraverse

A *functional concept* represents a global procedure, a sequence of subprocedures, or individual steps accomplishing a given task. For example, one of the questions asked for an exam was to explain the operation of Dijkstra's shortest path algorithm in plain text. For this question, the ideal answer contains 77 nodes and 80 arcs. The nodes are instances of concepts of the course ontology. Let us consider the algorithm's pseudocode, using some ontology concepts:

```
procedure Dijkstra
  While NOT AllVerticesAreVisited
    ChooseVertexV
    MarkVAsVisited
    ForEach Vertex W Adjacent to V
      ModifyDvOfW
      ModifyParentOfW
    end
  end
end
```

For every procedure or subprocedure, we create a corresponding functional concept: *Dijkstra*, *ChooseVertexV*, *MarkVAsVisited*, *ModifyDvOfW* and *ModifyParentOfW*. The functional concepts allow for a variable granularity of the algorithm description. When implementation details are needed, subprocedures (functional concepts) are in turn further decomposed. This decomposition only needs to be specified once and can then be reused, much like computer programming functions and procedures are defined once but can be called multiple times.

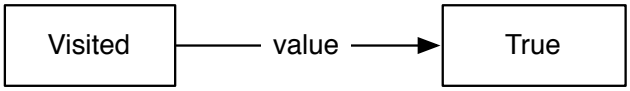
Functional concepts form only one category of concepts in our system's ontology. As mentioned, we define several other categories of concepts, which represent usual programming constructs. In particular, we give special consideration to the *flow structure* category, as it allows the temporal ordering of concept instances with two *sequential concepts*, *AndThen* and *OrThen*. These concepts allow instances of functional concepts to be ordered temporally. For example, *ChooseVertexV <AndThen> MarkVAsVisited* indicates that these concepts are required in that order. Alternatively, *ModifyDvOfW <OrThen> ModifyParentOfW* indicates that these concepts can be done in any order.

Another key concept category is the *operator*. This category defines the concept of logical negation (*NOT*), which allows modifying the type of iterative flow structures. For example, *While NOT AllVerticesAreVisited <do> X* can be transformed into *Until AllVerticesAreVisited <do> X*. It also allows specifying all conditions in the positive form (*VIsVisited*) and negating them easily (*NOT VIsVisited*).

The *variable* is another important concept category. These concepts are used to emulate variables of computer programming. Their treatment is discussed in Section 3.3.

In addition, we promoted relations, attributes and literals to full-fledged concepts (reified to first-class objects) in order to allow our algorithmic ontology to be as flexible and parametric as possible. This allows relations to be nested and to take not only concepts as their domain or image, but also other relations, attributes, and literals. We name these newly promoted concepts: *relation-concepts*, *attribute-concepts*, and *literal-concepts*. The concepts highlighted in italics in Tables II-1 and II-2 are relation-concepts. Figure II-2 shows an example of the use of attribute-concept *Visited* and literal-concept *True*, which are connected by the *value* relation.

Figure II-2 Example of attribute-concept and literal-concept in the semantic network



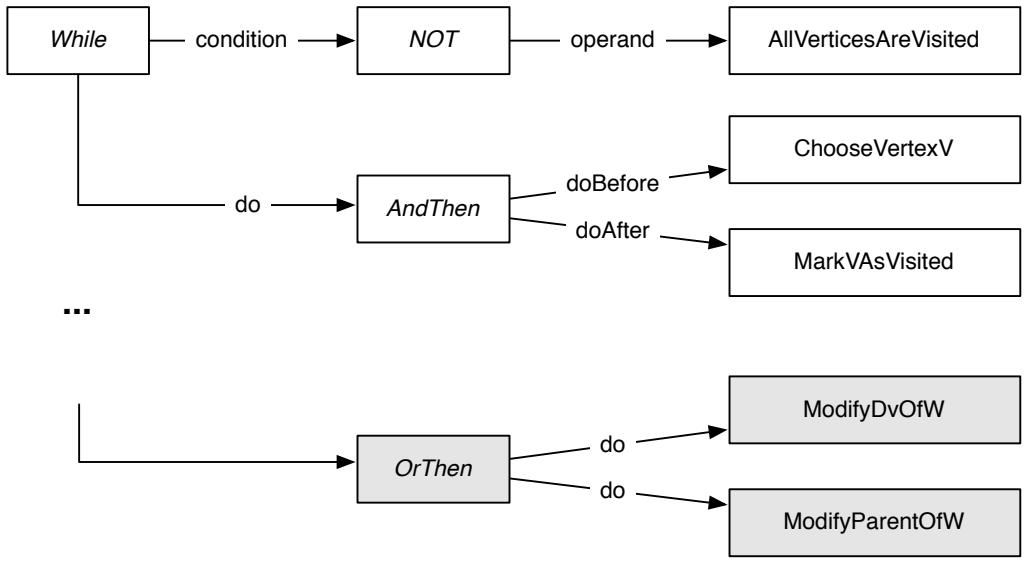
Using relation-concepts imposes the creation of new relations to connect all concepts uniformly. In the remainder of our paper, we simply refer to these new relations as *relations*, to distinguish them from relation-concepts. Our course ontology thus contains 15 relations (see Table II-2). The relations are used to label the edges of the semantic network, and hence participate only indirectly in the grading process. Figure II-3 shows an example of the nesting of relation-concepts (*While*, *NOT*, *AndThen*, *OrThen*) and of relations (*condition*, *operand*, *do*, *doBefore*, *doAfter*).

We now present four relations that need special consideration, as they are used for similarity measures and for the treatment of sequences of procedural knowledge. The *decomposition* relation is used in the semantic network to join a functional concept to the root of the subgraph detailing

Table II-2 Relations and example usage

Relation	**Example usage**
condition	*While* <condition> AllVerticesAreVisited
decomposition	ChooseVertexV <decomposition> *Assignment*
destination	Edge <destination> W (as in: edge from V to W)
do	*OrThen* <do> ModifyDvOfW
doAfter	*AndThen* <doAfter> MarkVAsVisited
doBefore	*AndThen* <doBefore> ChooseVertexV
field	*Dereference* <field> Visited (as in: V.Visited)
in	*ForEach* (<instance> V) <in> Vertices
instance	*ForAll* <instance> V
object	*Dereference* <object> V (as in: V.Visited)
operand	*Assignment* <operand> V (as in: V := . . .)
source	Edge <source> V (as in: edge from V to W)
suchAs	*ForAll* (<instance> V) <suchAs> Condition
to	Adjacent <to> V
value	Visited <value> True

Figure II-3 Example of relation-concepts and relations in the semantic network



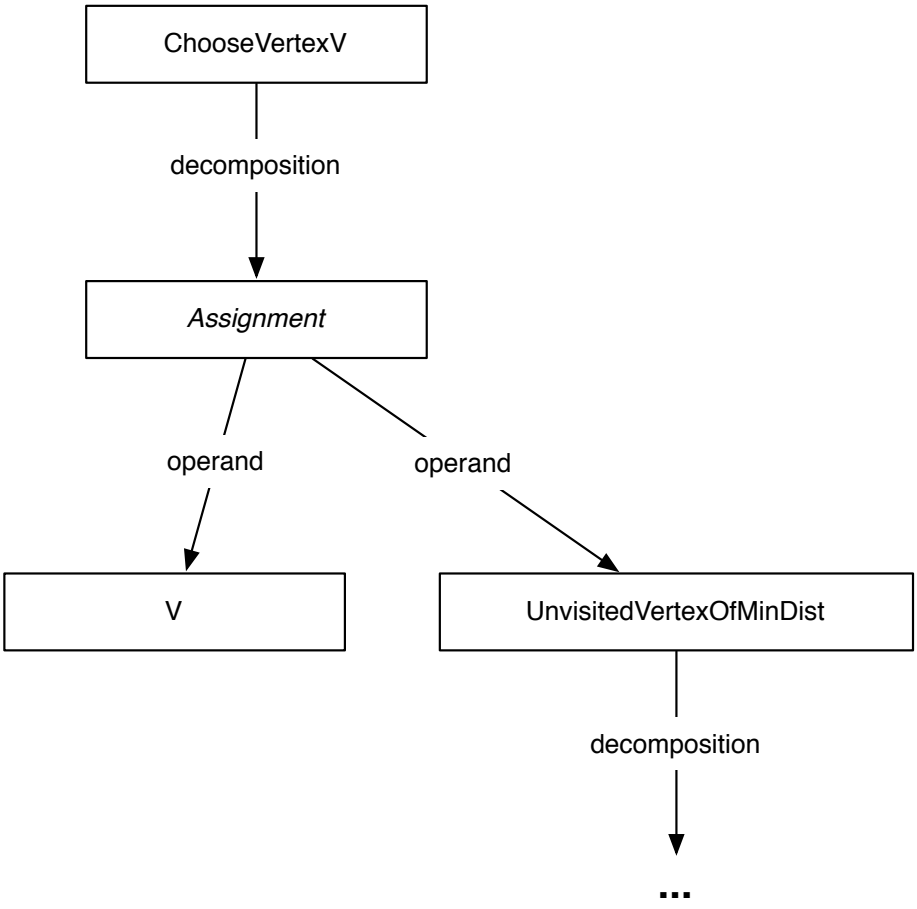
Relation-concepts are highlighted in italics.

it. It is used to zoom in on more details if needed in the student answer. For example, Figure II-4 shows how the *ChooseVertexV* functional concept is decomposed for Dijkstra's algorithm into the concepts *Assignment*, *V*, and *UnvisitedVertexOfMinDist*, which represents assigning to the variable *V* the unvisited vertex with total minimum distance. Further decomposition of the *UnvisitedVertex-OfMinDist* functional concept is not shown in this figure.

We have previously explained the behavior of *AndThen* and *OrThen*. The *doBefore* and *doAfter* relations are used in conjunction with the *AndThen* relation-concept to specify the required sequence. For example, Figure II-3 shows the use of *AndThen*, *doBefore*, and *doAfter* to specify that vertex *V* has to be chosen before it is marked as visited.

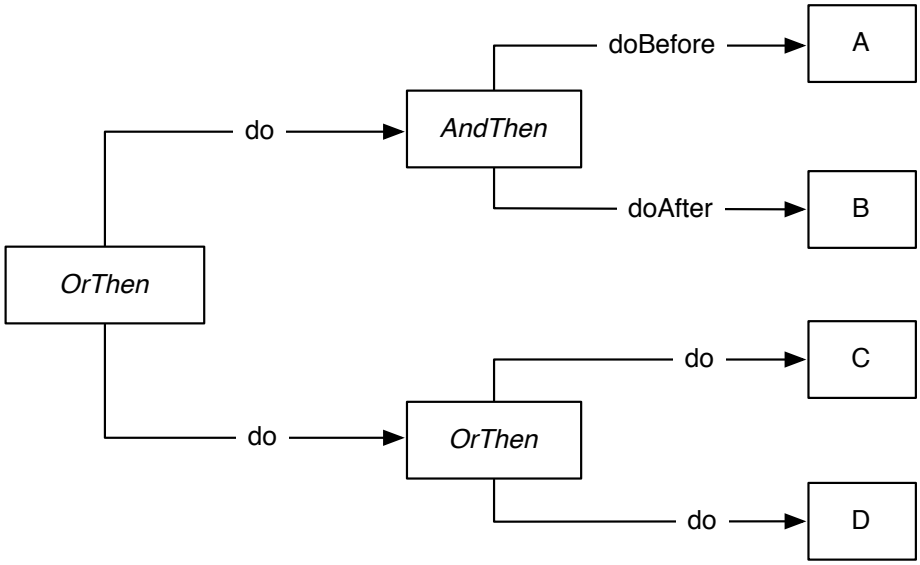
The *do* relation is used in conjunction with *OrThen* to indicate that two concepts can be performed in any order. For example, the grayed area in Figure II-3 illustrates the use of *OrThen* and *do* to indicate that the field *Dv* of the node *W* can be modified before or after changing its parent. The *do* relation is also used with the *While* relation-concept to indicate the block of steps to be repeated. For example, Figure II-3 shows the use of *While* and *do* to indicate that the subgraph rooted at *AndThen* has to be repeated.

Figure II-4 Example of the *decomposition* and *operand* relations in the semantic network



To specify more complex ordering relationships, *AndThen* and *OrThen* can be nested. Figure II-5 shows such nesting, which allows the four orderings: *ABCD*, *ABDC*, *CDAB*, and *DCAB*.

Figure II-5 Example of nesting *AndThen* and *OrThen* relation-concepts in the semantic network



3.2 Annotating Students' Answers

In the second phase, the system annotates a student's answer by using NLP techniques, namely: semantic networks, morphology analysis, synonymy and word sense disambiguation, tokenization, and stemming. It matches the concepts occurring in the ideal answer to words or expressions in the student's answer. This phase encompasses the *Preparation* and *Matching* steps.

The students enter their answer in a natural language (French, in our case). Our system could easily be adapted to support other languages by providing the appropriate dictionaries. The students are encouraged to produce full sentences; pseudocode and explicit code structure, such as indentation and braces, should be avoided in answers.

The system uses two constructed French dictionaries. The first (D1) is a thesaurus containing 244 synonyms [2]. The second (D2) contains 194 linguistic expressions associated to course ontology

2. See Annex G for the full thesaurus (in French).

concepts [3], and can be augmented as new expressions are discovered during annotation. The system recognizes 262 verbs, 370 nouns, 155 adjectives, 141 adverbs, and 48 other word categories, such as articles, prepositions, etc. It processes, through its morphology module, all expected word forms from the students.

3.2.1 Preparation

Since 60% of the students in our experiments are non-native French speakers, we manually spell-checked the answers and made grammatical revisions before processing the texts. The texts are then tokenized, stemmed, and synonyms are resolved to their canonical form using D1. The stemmed, canonical tokens are placed in a single array for each answer. We experimentally determined that our algorithm performs better when the answer's tokens are processed together, rather than one sentence at a time.

3.2.2 Matching

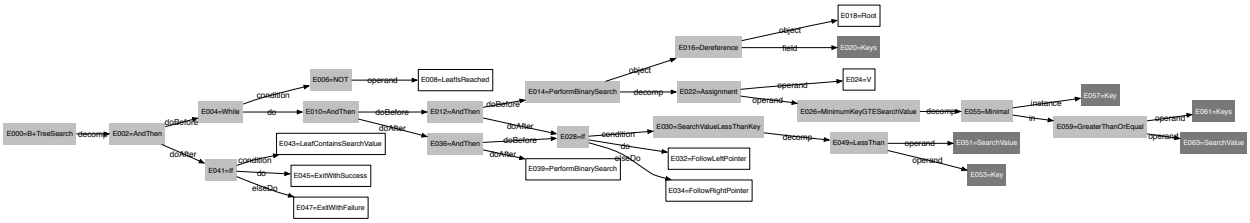
This step detects and matches the concepts of the teacher's ideal answer in the students' answers. In an ideal situation, the words and expressions in the students' answers are automatically annotated using previous annotations stored in the knowledge base, D2. In the case where the semantic network's concepts are not matched, the teacher has the option of adding an annotation, thus augmenting the system's knowledge base with the corresponding linguistic expression. These linguistic expressions are processed similarly to the student's answers (see *Preparation*, above). For example, when searching for concept *Parent*, previous linguistic expressions could be "its parent" and "the parent". Each of the stemmed, canonical tokens of the previous linguistic expressions are compared to the stemmed, canonical tokens of the student's answer. If a new linguistic expression is found in the student's answer, for example "the father node", then a new linguistic expression is added to the knowledge base. Section 3.4 shows how the *lexical similarity* measure takes into account the processing level needed to match the linguistic expressions.

---

3. See Annex H for the full ontological dictionary for $\mathcal{DIJ}$, Annex I for $\mathcal{DFS}$, and Annex J for $\mathcal{BT}$.

If an ideal answer's concept *C* is not found in the student's answer, the system looks for the ontological siblings of *C* first in the student's answer. If these concepts are not found, then the system looks for the parents, and finally the children of *C*. In case the matching fails, the concept *C* is marked as *absent*, otherwise it is marked as a *fuzzy match*. As will be seen in Section 3.4, fuzzy matches and absent concepts negatively impact the student's grade. Figure II-6 illustrates a student's semantic network overlay after the Matching phase. Fully or fuzzily matched concepts are highlighted in dark gray, while inferred concepts are highlighted in light gray. The remaining concepts are absent.

Figure II-6 Example of a student's semantic network overlay after the Matching phase



The matching algorithm adopts the strategy of "wait-and-see" regarding the problems of synonymy and ambiguity. A given concept can be found in multiple locations in the student's answer, sometimes with different linguistic expressions. Conversely, a linguistic expression in the answer can be matched when searching for different concepts of the ideal answer. For example, the token "and" in the student's answer is ambiguous as it can mean *AndThen* or *OrThen* in the ideal answer. The semantic network is used for disambiguation, as explained in the next phase.

3.3 Creating the Student's Semantic Network

The third phase links the annotated concepts (or relation-, attribute-, literal-concepts) of the student's answer to nodes of the ideal answer's semantic network. The result is the specific semantic network of the student's answer, which is an overlay of the ideal answer's semantic network. For example, consider the following sentence given by a student: "If W is closer than its parent V, the parent of W is modified to V." Figure II-7 shows a subset of the semantic network, corresponding to the phrase "parent of W".

Figure II-7 The semantic network corresponding to the phrase "parent of W"

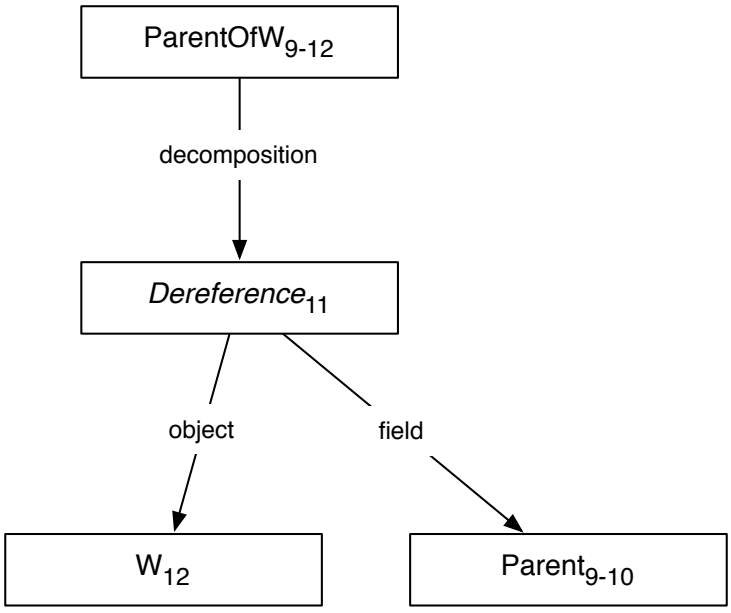


Table II-3 presents the annotations of the student's answer. The columns respectively indicate: the semantic network's concepts, the corresponding linguistic expressions, and their positions in the text. Note that the *ParentOfW* concept was not found in the student's text and hence is marked as absent. However, other concepts, such as *Parent* and *W*, were found more than once, and in different forms. This is an example of where a given concept can be found in multiple locations in the student's answer.

The third phase is performed by traversing the semantic network to find the parameters of every concept. We define the *parameters* of a given concept as the n-tuple including the concept itself and its children. Every parameter has zero or many corresponding linguistic expressions. Figure II-7 shows the parameters of the *Dereference* concept, which are (*Dereference*, *Parent*, *W*), corresponding to lines 2 to 4 of Table II-3. To find the parameters, our system relies on the relative positions of the linguistic expressions found in the student's text. It considers the distances between the candidate linguistic expressions of each parameter and selects those that optimize the textual proximity, i.e. that minimize the span of their corresponding positions.

Table II-3 Annotations of student's answer

Concepts	**Found linguistic expressions**	**Positions**
ParentOfW	*absent*	—
Parent	"its parent"	6–7
	"the parent"	9–10
Dereference	"of"	11
W	"W"	2
	"W"	12
V	"V"	8
	"V"	16
If	"If"	1
LessThan	"is closer than"	3–5
Assignment	"is modified to"	13–15

We define the *span* as the sum of the pair-wise distances among the candidate linguistic expressions. For example, the parameter *Parent* is matched to two candidate linguistic expressions, "its parent" and "the parent", at positions 6–7 and 9–10, respectively. Among the found positions of every parameter's linguistic expressions, the system selects the permutation having the minimum span. The *Dereference* parameter, corresponding to "of", has position 11. The *W* parameter is found at positions 2 and 12. The system chooses the sequence 9–10, 11, 12, because it has a zero span (the three linguistic expressions are contiguous). Therefore, it identifies "the parent", "of", and "W" as the linguistic expressions of the parameters of the *Dereference* node. In addition, the system stores in this node the calculated span (zero), the first and last found positions (9–12), and also the corresponding linguistic expression ("the parent of W"). These selected positions are indicated as subscripts in Figure II-7.

The system then continues up the semantic graph, identifying the parameters of the other concepts and linking them to the corresponding expressions, as seen before. This implies that recognized concepts become parameters of other concepts. For example, the parameters of the *ParentOfW* concept are (*ParentOfW*, *Dereference*). However, *ParentOfW* is absent in the student's text and only one candidate linguistic expression remains. Therefore, the system copies the information stored in the *Dereference* node into the *ParentOfW* node, which correctly inherits the same linguistic expression ("the parent of W").

While traversing the semantic network, the algorithm binds *variable concepts* as it discovers them in the student's answer. The system maintains a symbols table to keep track of the actual names used for the expected variables. If the student uses "X" and "Y" instead of the expected "V" and "W", the system normalizes it back to "V" and "W", in the case of Dijkstra's algorithm.

For example, consider the following sentence: "If node Y [=W] is closer than the parent node X [=V], its parent [the parent of V] is modified to X [=V]." Table II-4 gives the symbols table containing the bindings for this example answer. When a student uses a relative pronoun ("its parent", in our example), the referent is supposed to be the closest variable mentioned. In the example, "its parent" constitutes an ambiguous reference, and the referent is incorrectly determined to be "V".

Table II-4 Symbols table for example answer

Variable	Linguistic expression
V	"X"
W	"Y"

In general, the parameters have no specific order; therefore, the order of the linguistic expressions in the student answer is not taken into account. For example, note that *W*, *Dereference*, and *Parent*, cited in Figure II-7, could appear in any order in the student text, such as "parent of W" or "W has parent". An ordering is required with the *AndThen* relation-concept, where the parameter identified by the *doBefore* relation should come before the parameter identified by the *doAfter* relation, and also with asymmetrical operators (e.g. $a < b$).

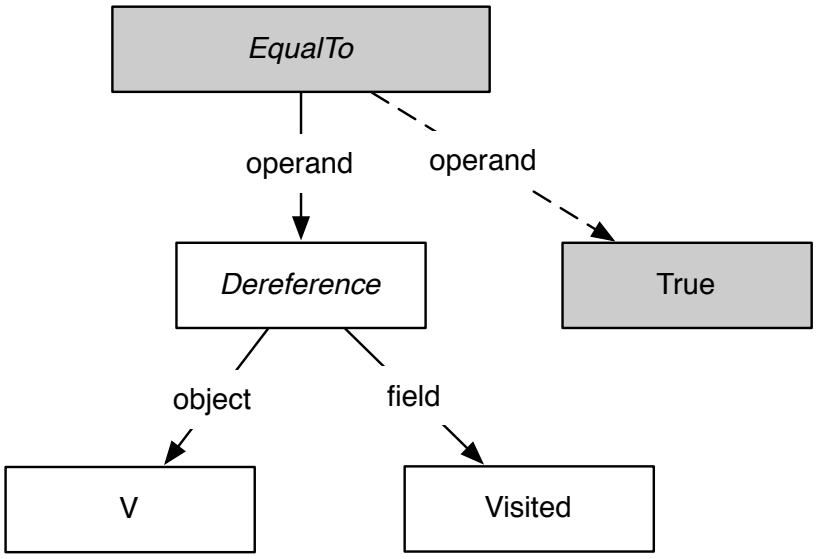
Note that this semantic representation is language-independent. It does not rely on grammatical patterns. In fact, in our implementation, the course ontology is in English and the student texts and linguistic expressions are in French. It is conceivable to adapt it easily to other languages by providing the linguistic expressions (and dictionaries) for those languages and linking them to the course ontology. This is possible thanks to the uniform representation of relations, attributes, and literals as concepts, and to a very granular course ontology that maximizes concepts' recognition in the texts.

Finally, the algorithm also infers "equals true" for all linguistic expressions of type "V is visited", as illustrated in Figure II-8, which represents one possible decomposition of the *VIsVisited* condition concept.

3.4 Grading Procedural Knowledge

The final phase consists of traversing the semantic network to calculate the similarity measures used by the grading algorithm. Our system assesses knowledge by five similarity measures: part-whole similarity (PS), textual proximity (TP), lexical similarity (LS), ontological proximity (OP), and temporal ordering (TO). All measures are in the $[0, 1]$ interval. PS recursively measures the

Figure II-8 The semantic network corresponding to the phrase "V is visited", where *EqualTo* operator concept and *True* literal concept are inferred



degree of detail given by the student in relation to the expected level of detail given by the teacher. TP measures the distance between related concepts in the student's answer by considering their position in the text. OP measures the distance between the expected concepts of the ideal answer and the found, inferred, and fuzzily matched concepts, using the course ontology. LS measures the distance between the linguistic expressions used by the student and the previously found linguistic expressions. Finally, TO compares the ordering of linguistic expressions in the student's answer to the temporal ordering of the ideal answer.

Teacher-provided weights allow adjusting the measures' contribution towards the students' grades, according to the teacher's preferences. In our case, we empirically selected a combination of these weights that gives grades close to those given by human graders, with a positive correlation. This combination showed better results than equal weighting for all the measures. We used the same combination for all students' answers to all questions. The process of finding the best combination could be automated if desired.

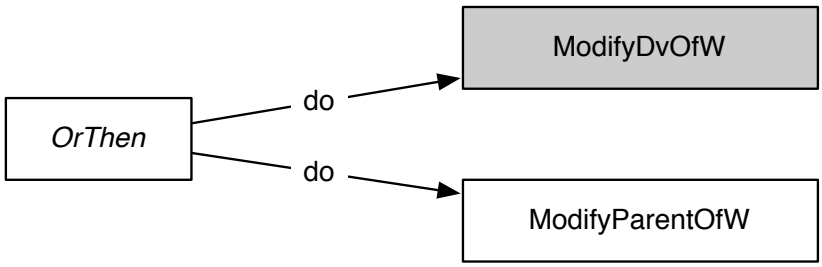
3.4.1 Part-Whole Similarity

The part-whole similarity measure is based on the assumption that the decomposition of a concept into its subconcepts is a kind of *isPartOf* relation. We further assume that this relation is transitive, and give the measure a recursive definition. A leaf of the ideal answer matched in the student answer scores 1.0, and an unmatched leaf (marked absent in phase two) scores zero. For each node c_i in the remaining nodes (i.e. the internal nodes and the root), the part-whole similarity, $PS(c_i)$, is the ratio of the number of matched nodes in the subgraph rooted at c_i to the total number of nodes therein, as shown in Equation II-1.

$$PS(c_i) = \frac{\sum_{c_j \in children(c_i)} PS(c_j)}{|children(c_i)|} \tag{II-1}$$

For example, consider Figure II-9, where the *OrThen* concept has two children: *ModifyDvOfW*, which has been matched; and *ModifyParentOfW*, which is absent. Therefore, only half of the child nodes have been matched. The PS measure for the *ModifyDvOfW* node is 1.0 and that of *ModifyParentOfW* is zero. Therefore, the PS measure of the *OrThen* node is: $(1 + 0)/2 = 0.5$.

Figure II-9 Example of partially matched subgraph in the semantic network



Matched concept is highlighted in gray.

For a student answer R and an ideal answer I, the total part-whole similarity, $PS(R, I)$, is calculated using Equation II-2, where $concepts(I)$ are the concept nodes of the ideal answer. Note that since R is an overlay of I, the summation only takes into account the concepts of I.

$$PS(R, I) = \frac{\sum_{c_i \in concepts(I)} PS(c_i)}{|concepts(I)|} \quad \text{(II-2)}$$

3.4.2 Textual Proximity

This measure is inversely proportional to the distance between the parameters of a concept (see Section 3.3) in a student text. The TP measure of a concept node c_i of the ideal answer, $TP(c_i)$, is 1.0 when the corresponding linguistic expressions of the node's parameters are contiguous in the student text. It is defined as zero if no parameters are found in the student text for concept c_i. Otherwise, it is calculated by Equation II-3. Note that parameters separated by unrecognized or irrelevant words increase the span, thus lowering the TP measure.

$$TP(c_i) = 1 - \frac{avgspan(c_i)}{|tokens|} \quad \text{(II-3)}$$

In Equation II-3, $avgspan(c_i)$ is the average span of all occurrences of concept c_i in the semantic network. In order to give a value in the $[0, 1]$ interval, the span is normalized by dividing by the number of tokens in the student answer, $|tokens|$. For a student answer R and an ideal answer I, the total textual proximity, $TP(R, I)$, is calculated using Equation II-4.

$$TP(R, I) = \frac{\sum_{c_i \in concepts(I)} TP(c_i)}{|concepts(I)|} \quad \text{(II-4)}$$

3.4.3 Lexical Similarity

This measure is similar to OeLE's *linguistic similarity* measure, which uses the Levenshtein distance. Our measure is calculated using the the resemblance between two linguistic expressions, $L(e_i, e_j)$. The first linguistic expression, e_i, is taken from the student answer, and the second, e_j, from the dictionary (D2) containing linguistic expressions. If the two expressions are identical, $L(e_i, e_j) = 1.0$. If they share the same radical (e.g. "node" and "nodes"), then $L(e_i, e_j) = 0.6$. If one is a synonym of the other (e.g. "node" and "vertex"), then $L(e_i, e_j) = 0.3$. Otherwise (e.g.

"node" and "graph"), $L(e_i, e_j)$ is zero. If the linguistic expressions contain more than one token, the L measure is the maximum of the individual L values.

During annotation, concept nodes of the ideal answer's semantic network are searched in the student answer (see Section 3.2.2). For each concept c_i, the lexical similarity, $LS(c_i)$, is defined by taking the maximum L value obtained while matching the linguistic expressions of D2 associated to concept c_i to the student answer's linguistic expressions. For a student answer R and an ideal answer I, the total lexical similarity, $LS(R, I)$, is calculated using Equation II-5.

$$LS(R, I) = \frac{\sum_{c_i \in concepts(I)} LS(c_i)}{|concepts(I)|} \tag{II-5}$$

3.4.4 Ontological Proximity

This measure is introduced to take into account the fact that a concept c_i in the ideal answer can be fully or fuzzily matched to a concept in the student answer. In case concept c_i is absent in the student answer, but some of its children are matched, c_i is *matched by inference*. For each concept c_i of the ideal answer, the OP measure, $OP(c_i)$, is calculated according to the following rules. If concept c_i is absent, $OP(c_i)$ is zero. If concept c_i is fully matched, $OP(c_i) = 1.0$. If c_i is matched by inference, $OP(c_i) = 0.5$.

Otherwise, the concepts are fuzzily matched, and the OP measure is calculated by using OeLE's *concept proximity* measure, written $CP(c_i, c_j)$ below. This measure evaluates the distance between concept c_i of the ideal answer and the concept c_j fuzzily matched in the student answer. The latter could be a sibling, an ancestor or a child in the course ontology (see Section 3.2). The CP measure relies on the class hierarchy defined by OWL's *subClassOf* relation, and is defined as zero if concepts c_i and c_j have no taxonomic parent.

$$CP(c_i, c_j) = 1 - \frac{|nodes(c_i, c_j)|}{|concepts|} \tag{II-6}$$

In Equation II-6, $|nodes(c_i, c_j)|$ is the number of concepts separating c_i and c_j through the shortest path in the class hierarchy, and $|concepts|$ is the total number of concepts in the course ontology [8]. A shorter path thus indicates a stronger similarity between the two concepts.

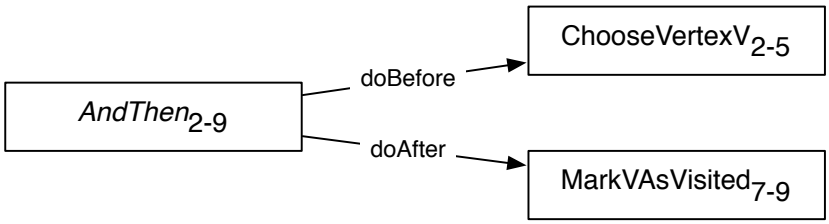
For a student answer R and an ideal answer I, the total ontological proximity, $OP(R, I)$, is calculated using Equation II-7.

$$OP(R, I) = \frac{\sum_{c_i \in concepts(I)} OP(c_i)}{|concepts(I)|} \tag{II-7}$$

3.4.5 Temporal Ordering

This measure takes into account the ordering specified by the *AndThen* concepts in the ideal answer. For a student answer R, the TO measure, $TO(R)$, is calculated by dividing the total number of *AndThen* concepts having the right ordering of linguistic expressions in the student answer by the total number of *AndThen* concepts in the ideal answer. Figure II-10 gives an example of temporal ordering with the *AndThen* sequential flow structure concept in the ideal answer. We define the *right order* like so: the linguistic expression corresponding to the subgraph indicated by the *doBefore* relation (*ChooseVertexV*, in the example, at positions 2–5) should appear before the linguistic expression corresponding to the subgraph indicated by the *doAfter* relation (*MarkVAsVisited*, at positions 7–9).

Figure II-10 Example of temporal ordering with *AndThen* in the semantic network



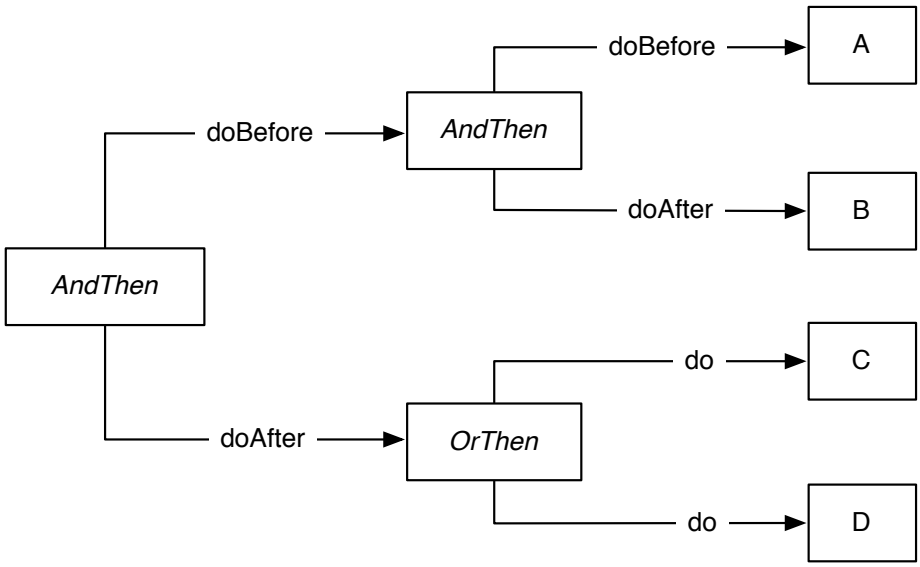
Corresponding positions of linguistic expressions are indicated as subscripts.

For a student answer R and an ideal answer I, the temporal ordering, $TO(R, I)$, is calculated using Equation II-8.

$$TO(R, I) = \frac{\text{number of } AndThen \text{ concepts in right order in } R}{\text{total number of } AndThen \text{ concepts in } I} \quad \text{(II-8)}$$

As an additional example, Table II-5 shows temporal ordering values for the all possible answer permutations, given the semantic network of Figure II-11, where these two permutations are fully correct: *ABCD* and *ABDC*. Note that overlapping linguistic expressions are not considered to be in the right order, so the permutation *ACBD*, for example, does not satisfy the root *AndThen* condition (since *AB* is found at position 1–3, and *CD* at position 2–4).

Figure II-11 Nested *AndThen* and *OrThen* relation-concepts in the semantic network



3.4.6 Grading

To obtain the final grade, $G(R, I)$ of a student answer R, considering ideal answer I, we use Equation II-9 below. All the measures mentioned above are calculated for each concept (or relation-, attribute-, literal-concept) c_i of the ideal answer's semantic network and are weighted to become the terms of our equation.

Table II-5 Temporal ordering values for all answer permutations

Permutation	***A* before *B***	***AB* before *CD***	**Temporal Ordering**
ABCD	✓	✓	1.0
ABDC	✓	✓	1.0
ACBD	✓	×	0.5
ACDB	✓	×	0.5
ADBC	✓	×	0.5
ADCB	✓	×	0.5
BADC	×	✓	0.5
BACD	×	✓	0.5
BCDA	×	×	0.0
BCAD	×	×	0.0
BDCA	×	×	0.0
BDAC	×	×	0.0
CABD	✓	×	0.5
CADB	✓	×	0.5
CBAD	×	×	0.0
CBDA	×	×	0.0
CDAB	✓	×	0.5
CDBA	×	×	0.0
DACB	✓	×	0.5
DABC	✓	×	0.5
DBCA	×	×	0.0
DBAC	×	×	0.0
DCBA	×	×	0.0
DCAB	✓	×	0.5

$$G(R,I) = \left\{ \begin{array}{l} \quad ps \times PS(R,I) \\ + tp \times TP(R,I) \\ + ls \times LS(R,I) \\ + op \times OP(R,I) \\ + to \times TO(R,I) \end{array} \right\} \quad \text{(II-9)}$$

In Equation II-9, ps, tp, ls, op, and to are weights in the $[0,1]$ interval corresponding to the respective measures. The combination of weights used in our experiments is: $ps = 0.16$, $tp = 0.16$, $ls = 0.06$, $op = 0.39$, and $to = 0.23$.

4 Experimental Results

To evaluate our system, we used questions and answers from three exams on computer algorithms given at Université de Moncton between 2011 and 2013. The three exam questions [4] respectively asked to describe the detailed operation of Dijkstra's shortest path algorithm ($\mathcal{DIJ}$), depth-first search ($\mathcal{DFS}$), and B+ tree search ($\mathcal{BT}$). Each question was answered respectively by 13, 22, and 18 students. The average length of students' answers was respectively 91, 40, and 82 words, with a minimum of 8 words and a maximum of 161 [5]. The system's grading results were respectively compared to grades supplied by 7, 4, and 4 instructors and graduate students. The average, standard deviation, and Pearson correlation were used for this comparison. Regression analysis was performed on the data to determine the statistical significance.

The course ontology contains 114 concepts and 17 relations [6], shared by the three questions. The semantic networks [7] for $\mathcal{DIJ}$, $\mathcal{DFS}$, and $\mathcal{BT}$ respectively contain 77, 9, and 32 concepts (nodes) and 80, 9, and 32 relations (arcs). The reuse of ontology concepts facilitates the adoption of new questions. For example, the grading of a question about breadth-first search would not require adding concepts to the ontology. The semantic network would reflect the new order of node

4. See Annex C for the original texts in French and their translations.

5. We do not disclose the students' answers here, for confidentiality reasons. The students' identity was also not revealed to the human graders.

6. See Annex B for the full course ontology.

7. See Annex D for $\mathcal{DIJ}$, Annex E for $\mathcal{DFS}$, and Annex F for $\mathcal{BT}$.

traversal. However, a question about the B+ tree would use those concepts from a binary search tree, but these concepts might not be sufficient to represent the whole semantic network.

For $\mathcal{DIJ}$, Table II-6 shows the grades, over 4 points, given by human graders (G1–G7) for each student (S1–S13). The instructors gave average grades between 1.92 and 3.31, with an average of 2.80 and a standard deviation of 0.78.

Table II-6 Grades given by human graders for $\mathcal{DIJ}$

Student	**G1**	**G2**	**G3**	**G4**	**G5**	**G6**	**G7**	**Avg-Human**	**Stdev**
S1	3	1	2.8	2	1	2.5	2	2.0	0.8
S2	4	2	3.6	4	4	3.25	3	3.4	0.7
S3	4	2	3.6	4	4	4	4	3.7	0.8
S4	3	1	3.6	3	3	2.5	2	2.6	0.9
S5	4	3	3.6	3.5	2	3.75	4	3.4	0.7
S6	3	1	2.4	2.5	1	2	1	1.8	0.8
S7	4	3	4	4	4	3.75	3	3.7	0.5
S8	3.5	2	3.2	3	3	3.25	4	3.1	0.6
S9	1	2	2	1	2	1.25	3	1.8	0.7
S10	4	3	3.2	3	3	3.75	3	3.3	0.4
S11	4	1	3.6	2.5	1	2.75	2	2.4	1.2
S12	1.5	1	2	2.5	0	2.5	2	1.6	0.9
S13	4	3	3.2	4	3	3.75	4	3.6	0.5
Avg	3.3	1.9	3.1	3.0	2.4	3.0	2.9	2.8	—
Stdev	1.0	0.9	0.7	0.9	1.3	0.8	1.0	0.8	—

Table II-7 shows a comparison of the average human grades for each student with those of the system. The system gave an average of 3.11 points (out of 4), with a standard deviation of 0.38. Note that the average grade assigned by the system falls in 77% of cases within a distance of one standard deviation from the average of the human graders (Avg-Human). Moreover, the correlation between Avg-Human and the system's grades is 0.70 ($p < 0.05$), which indicates a positive relationship.

Table II-7 Comparison of average human grades with those of PROCMARK for $\mathcal{DIJ}$

Student	**Avg-Human**	**PROCMARK**	**Difference**
S1	2.04	2.42	0.38
S2	3.41	3.55	0.15
S3	3.66	3.34	−0.31
S4	2.59	3.03	0.44
S5	3.41	3.31	−0.09
S6	1.84	2.34	0.50
S7	3.68	3.17	−0.51
S8	3.14	3.14	0.00
S9	1.75	2.79	1.04
S10	3.28	3.50	0.22
S11	2.41	3.37	0.96
S12	1.64	3.13	1.49
S13	3.56	3.33	−0.24
Avg	2.80	3.11	0.31
Stdev	0.78	0.38	0.58

Figure II-12 compares the grades given by human graders to those of the system. Vertical bars indicate the minimum and maximum grades given to this student by the human graders. Figure II-13 shows the correlation of the average human grades with those given by the system.

Figure II-12 Comparison of human grades with those of PROCMARK for $\mathcal{DIJ}$

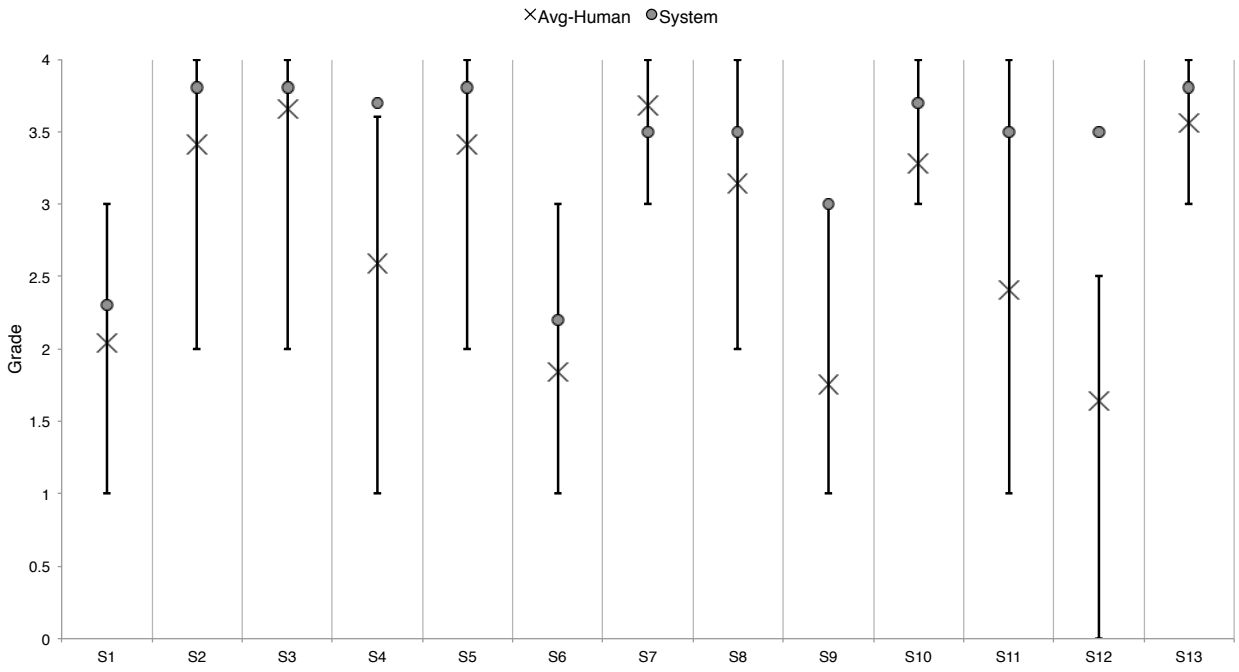


Table II-8 shows the correlations between pairs of human graders, as well as the correlations between individual graders and the system. Minimum and maximum values between human graders are highlighted in bold, as well as the maximum correlation between human and system grades.

Figure II-14 shows the grades given by human graders in ascending order from left to right. Vertical bars indicate the minimum and maximum grades given to this student by the individual human graders. We can observe that the system gave more generous grades than the humans, in all but one case (S7). Figure II-15 shows the correlation of the individual measures (PS, TP, LS, OP, and TO) with the average grade given by human graders, and with the final grade assigned by the system. The TO measure has a very poor correlation (0.18) with Avg-Human. This could be explained by the complex nature of the algorithm (nested *for* loops), which is difficult both to express and to assess in free-text answers without proper indentation or braces.

Figure II-13 Correlation of human grades with those of PROCMARK for $\mathcal{DIJ}$

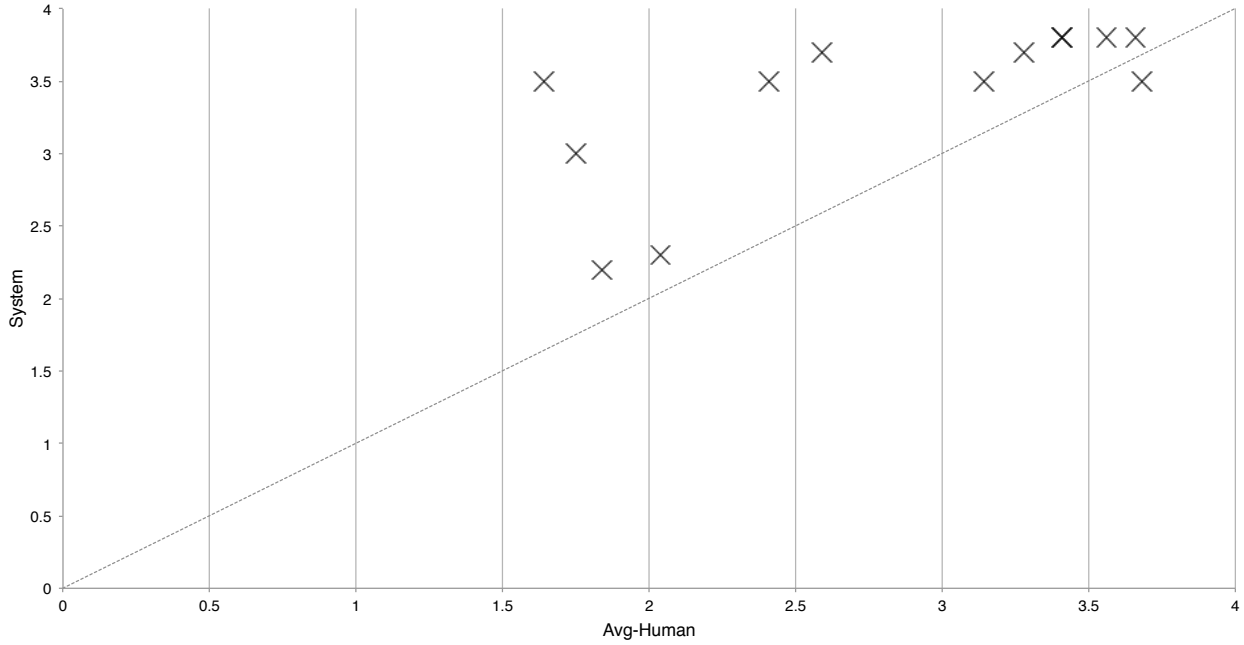


Table II-8 Correlations between pairs of human graders and PROCMARK for $\mathcal{DIJ}$

	G1	**G2**	**G3**	**G4**	**G5**	**G6**	**G7**	**PROCMARK**
G1	1.00	0.46	0.87	0.79	0.56	0.83	**0.39**	0.53
G2		1.00	0.41	0.53	0.61	0.67	0.77	0.54
G3			1.00	0.76	0.69	0.74	0.40	0.58
G4				1.00	0.69	**0.88**	0.51	0.63
G5					1.00	0.61	0.62	0.53
G6						1.00	0.66	**0.70**
G7							1.00	0.60
PROCMARK								1.00

Figure II-14 Ranking of human grades compared to PROCMARK for $\mathcal{DIJ}$

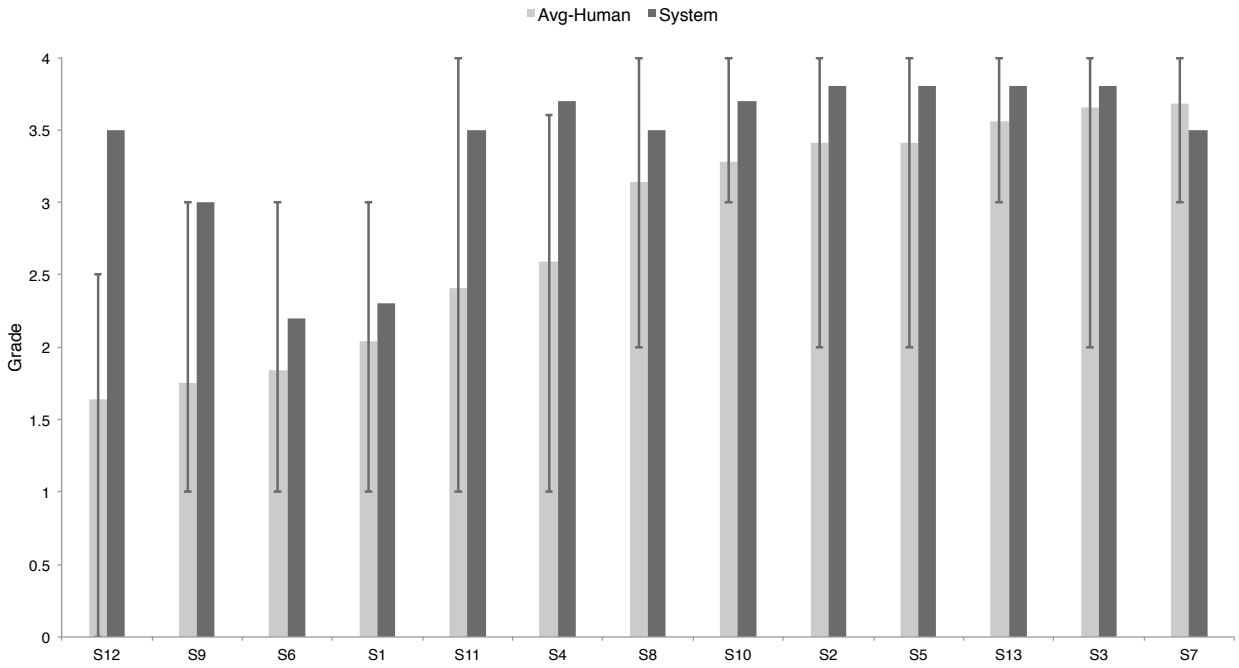
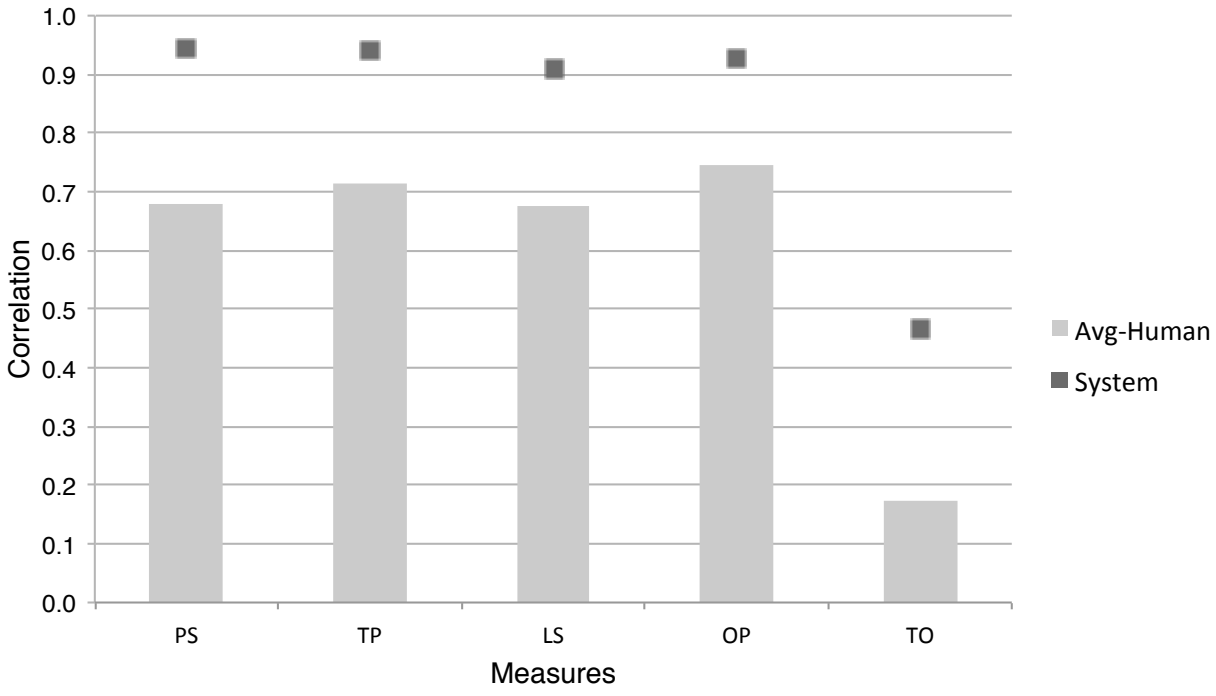


Figure II-15 Correlation of individual measures for $\mathcal{DIJ}$



For DFS, Table II-9 shows the grades, over 5 points, given by human graders (G1–G4) for each student (S1–S22). The instructors gave average grades between 2.36 and 3.16, with an average of 2.81 and a standard deviation of 1.45.

Table II-10 shows a comparison of the average human grades for each student with those of the PROCMARK system. The system gave an average of 3.13 points (out of 5), with a standard deviation of 1.23. Note that the average grade assigned by the system falls in 50% of cases within a distance of one standard deviation from Avg-Human. The correlation between the average human grades and those of the system is 0.79 ($p < 0.05$), which indicates a positive relationship.

Figure II-16 compares the grades given by human graders to those of the system. Figure II-17 shows the correlation of the average human grades with those given by the system.

Figure II-16 Comparison of human grades with those of PROCMARK for $\mathcal{DFS}$

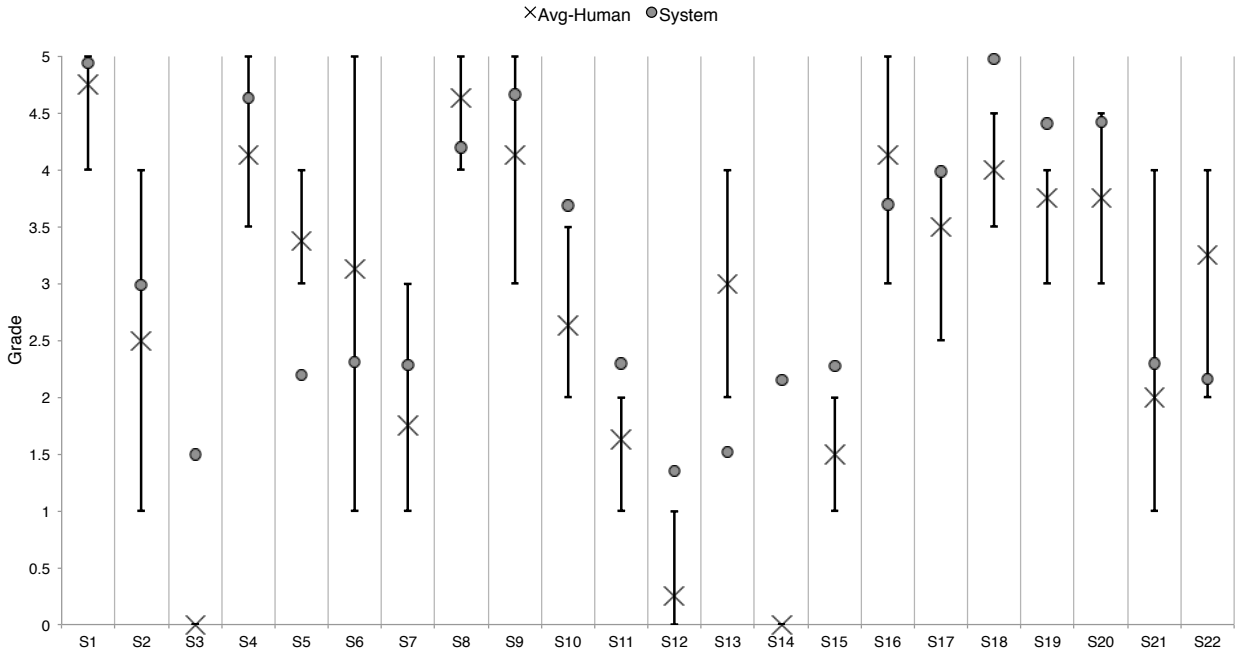


Table II-11 shows the correlations between pairs of human graders. The correlations between individual graders and the system are also given. Minimum and maximum values between human graders are highlighted in bold, as well as the maximum correlation between human and system grades.

Table II-9 Grades given by human graders for $\mathcal{DFS}$

Student	**G1**	**G2**	**G3**	**G4**	**Avg-Human**	**Stdev**
S1	5	4	5	5	4.75	0.50
S2	3	4	2	1	2.50	1.29
S3	0	0	0	0	0.00	0.00
S4	3.5	5	4	4	4.13	0.63
S5	3	4	3.5	3	3.38	0.48
S6	2.5	4	5	1	3.13	1.75
S7	1	3	2	1	1.75	0.96
S8	5	4.5	5	4	4.63	0.48
S9	4.5	5	4	3	4.13	0.85
S10	2	3	3.5	2	2.63	0.75
S11	1	2	1.5	2	1.63	0.48
S12	1	0	0	0	0.25	0.50
S13	2.5	4	3.5	2	3.00	0.91
S14	0	0	0	0	0.00	0.00
S15	1	1	2	2	1.50	0.58
S16	4.5	3	5	4	4.13	0.85
S17	2.5	4	4	3.5	3.50	0.71
S18	4.5	4	4	3.5	4.00	0.41
S19	4	4	4	3	3.75	0.50
S20	4.5	3	4.5	3	3.75	0.87
S21	1	4	1	2	2.00	1.41
S22	2	4	4	3	3.25	0.96
Avg	2.64	3.16	3.07	2.36	2.81	—
Stdev	1.63	1.57	1.71	1.42	1.45	—

Table II-10 Comparison of average human grades with those of PROCMARK for $\mathcal{DFS}$

Student	**Avg-Human**	**PROCMARK**	**Difference**
S1	4.75	4.94	0.19
S2	2.50	2.99	0.49
S3	0.00	1.50	1.50
S4	4.13	4.62	0.50
S5	3.38	2.19	– 1.18
S6	3.13	2.30	– 0.82
S7	1.75	2.28	0.53
S8	4.63	4.20	– 0.43
S9	4.13	4.66	0.54
S10	2.63	3.69	1.06
S11	1.63	2.30	0.67
S12	0.25	1.35	1.10
S13	3.00	1.52	– 1.48
S14	0.00	2.15	2.15
S15	1.50	2.27	0.77
S16	4.13	3.70	– 0.43
S17	3.50	3.98	0.48
S18	4.00	4.97	0.97
S19	3.75	4.40	0.65
S20	3.75	4.41	0.66
S21	2.00	2.29	0.29
S22	3.25	2.16	– 1.09
Avg	2.81	3.13	0.32
Stdev	1.45	1.23	0.90

Figure II-17 Correlation of human grades with those of PROCMARK for $\mathcal{DFS}$

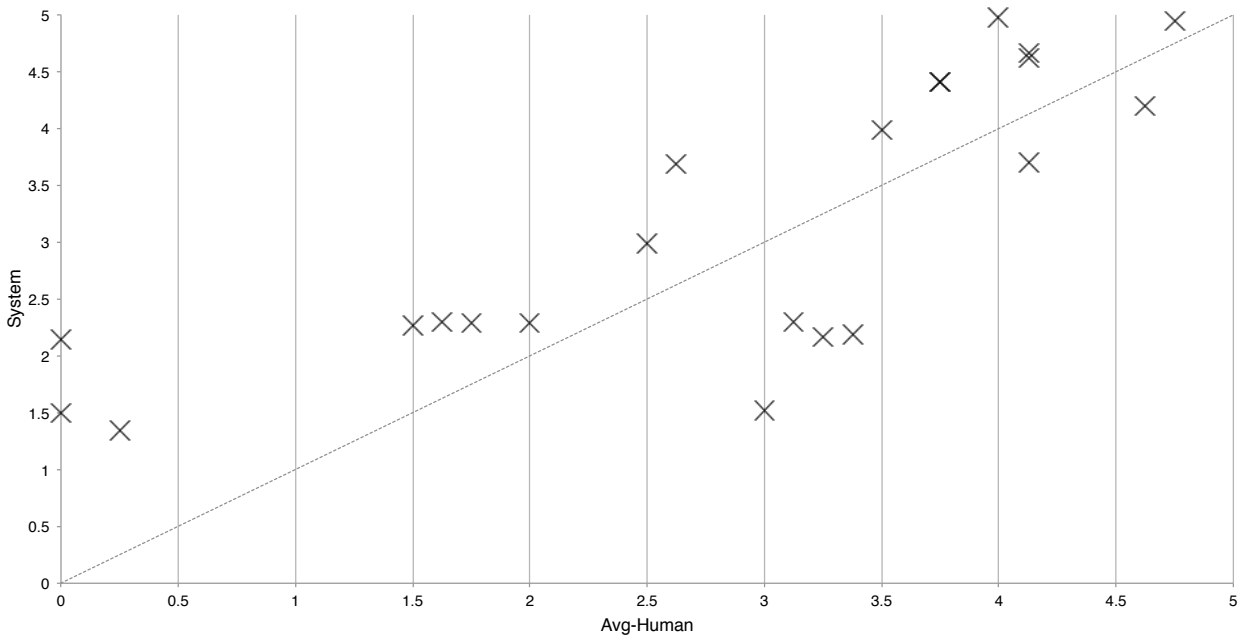


Table II-11 Correlations between pairs of human graders and PROCMARK for $\mathcal{DFS}$

	G1	**G2**	**G3**	**G4**	**PROCMARK**
G1	1.00	0.71	**0.85**	0.82	**0.83**
G2		1.00	0.77	**0.70**	0.59
G3			1.00	0.81	0.68
G4				1.00	0.77
PROCMARK					1.00

Figure II-18 shows the grades given by human graders in ascending order from left to right. We can observe that the system gave more generous grades than humans in the zero to 2.5 grade range (out of 5), and less generous grades than humans in the 2.5 to 3.5 range. Two of the students (S3 and S14) were given grades of zero by all human graders; however, the system gave them 1.50 and 2.15 points respectively. Student S3 had the shortest answer (for all three algorithms), counting only 8 words, and student S14 answered in only 14 words [8]. The system might have overestimated the value of the keywords present in both answers.

Figure II-18 Ranking of human grades compared to PROCMARK for $\mathcal{DFS}$

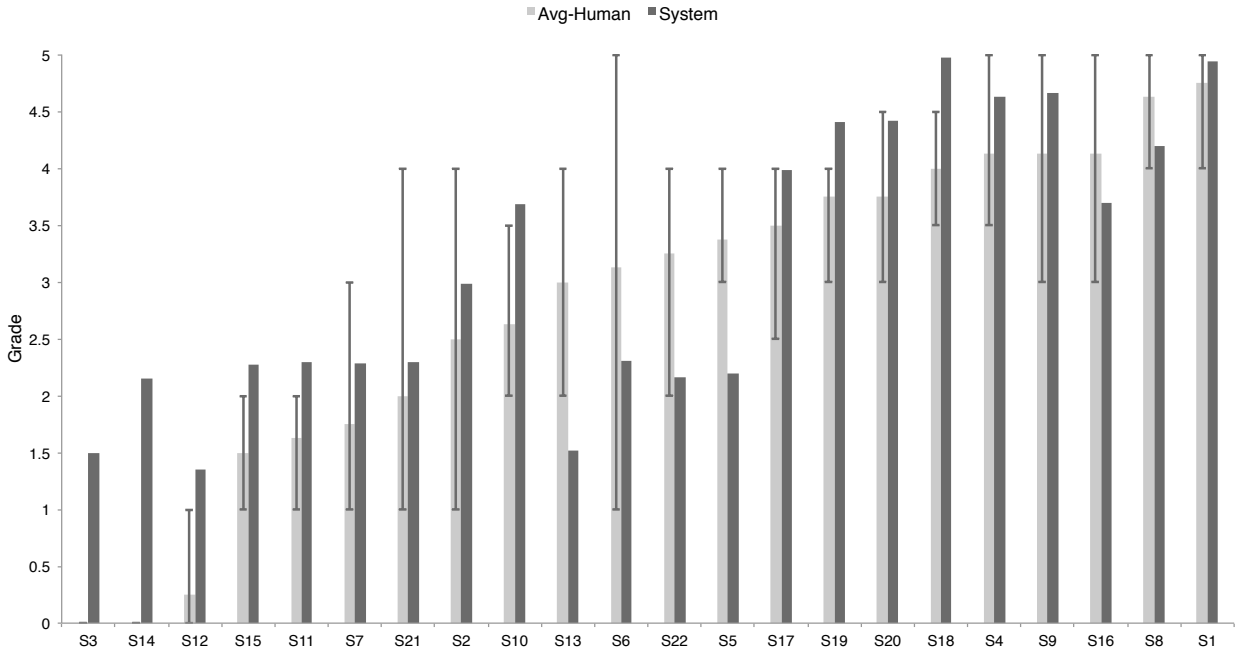


Figure II-19 shows the correlation of the individual measures with the average grade given by human graders, and with the final grade assigned by the system. All measures perform equally well with this algorithm.

For BT, Table II-12 shows the grades, over 10 points, given by human graders (G1–G4) for each student (S1–S18). The instructors gave average grades between 5.28 and 6.39, with an average of 5.98 and a standard deviation of 2.27.

8. Recall that the average length of student answers for this algorithm was 40 words.

Table II-12 Grades given by human graders for $\mathcal{BT}$

Student	**G1**	**G2**	**G3**	**G4**	**Avg-Human**	**Stdev**
S1	8.5	8	7	8	7.88	0.63
S2	9	8	7.5	8.5	8.25	0.65
S3	5	1	5	6	4.25	2.22
S4	8	5	5	6	6.00	1.41
S5	6	9	8	8.5	7.88	1.31
S6	10	10	7	8.5	8.88	1.44
S7	2.5	5	2	2	2.88	1.44
S8	9	9	7.5	7	8.13	1.03
S9	10	9	9	10	9.50	0.58
S10	5.5	6	4	7	5.63	1.25
S11	5	5	7	5	5.50	1.00
S12	3	4	2	3	3.00	0.82
S13	3.5	4	4	3	3.63	0.48
S14	4	5	2	5	4.00	1.41
S15	7	10	7	8.5	8.13	1.44
S16	5.5	8	5	7.5	6.50	1.47
S17	3	8	5	5	5.25	2.06
S18	3.5	1	1	4	2.38	1.60
Avg	6.00	6.39	5.28	6.25	5.98	—
Stdev	2.55	2.81	2.39	2.29	2.27	—

Figure II-19 Correlation of individual measures for $\mathcal{DFS}$

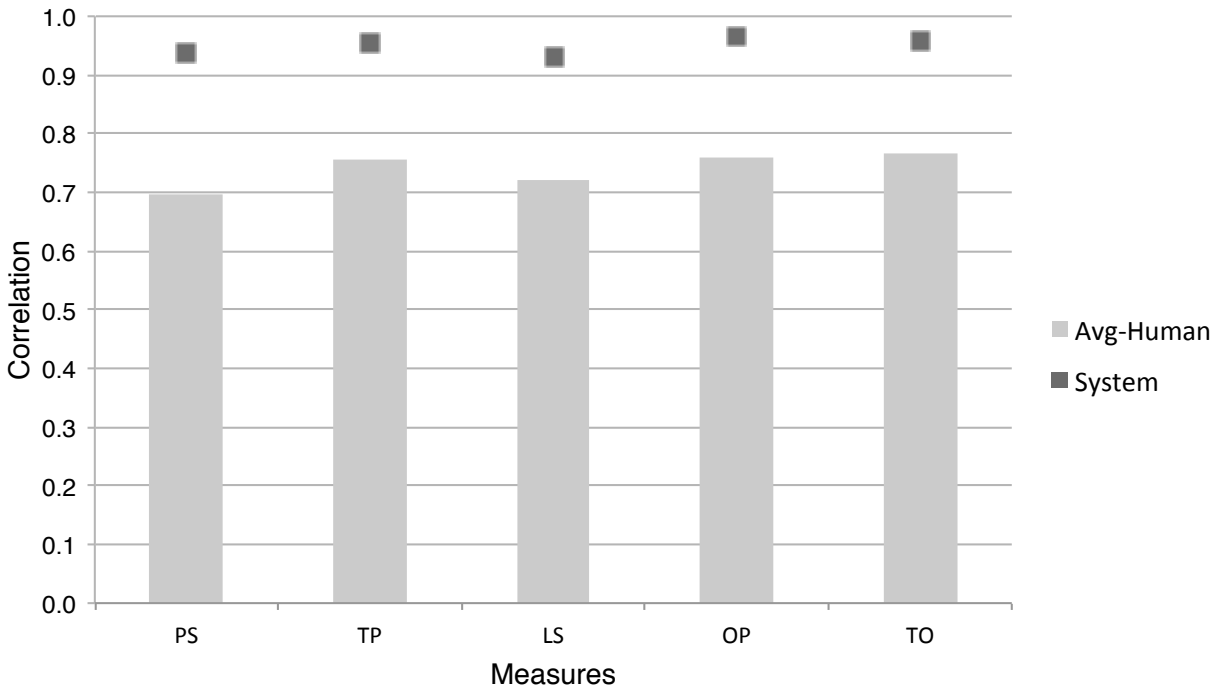


Table II-13 shows a comparison of the average human grades for each student with those of the system. The system gave an average of 5.94 points (out of 10), with a standard deviation of 1.17. Note that the average grade assigned by the system falls in 44% of cases within a distance of one standard deviation from Avg-Human. The correlation between the average results of human graders and those of the system is 0.79 ($p < 0.05$).

Figure II-20 compares the grades given by human graders to those of the system. Figure II-21 shows the correlation of the average human grades with those given by the system.

Table II-14 shows the correlations between pairs of human graders. The correlations between individual graders and the system are also given. Minimum and maximum values between human graders are highlighted in bold, as well as the maximum correlation between human and system grades.

Figure II-22 shows the grades given by human graders in ascending order from left to right. We can observe that the system gave again more generous grades than humans in the zero to 4 grade

Table II-13 Comparison of average human grades with those of PROCMARK for $\mathcal{BT}$

Student	**Avg-Human**	**PROCMARK**	**Difference**
S1	7.88	7.20	−0.68
S2	8.25	7.26	−0.99
S3	4.25	5.00	0.75
S4	6.00	5.29	−0.71
S5	7.88	6.26	−1.62
S6	8.88	7.74	−1.14
S7	2.88	5.13	2.25
S8	8.13	6.25	−1.88
S9	9.50	7.77	−1.73
S10	5.63	5.95	0.33
S11	5.50	4.79	−0.71
S12	3.00	5.01	2.01
S13	3.63	4.97	1.34
S14	4.00	5.75	1.75
S15	8.13	7.65	−0.47
S16	6.50	5.77	−0.73
S17	5.25	3.77	−1.48
S18	2.38	5.39	3.01
Avg	5.98	5.94	−0.04
Stdev	2.27	1.17	1.52

Figure II-20 Comparison of human grades with those of PROCMARK for $\mathcal{BT}$

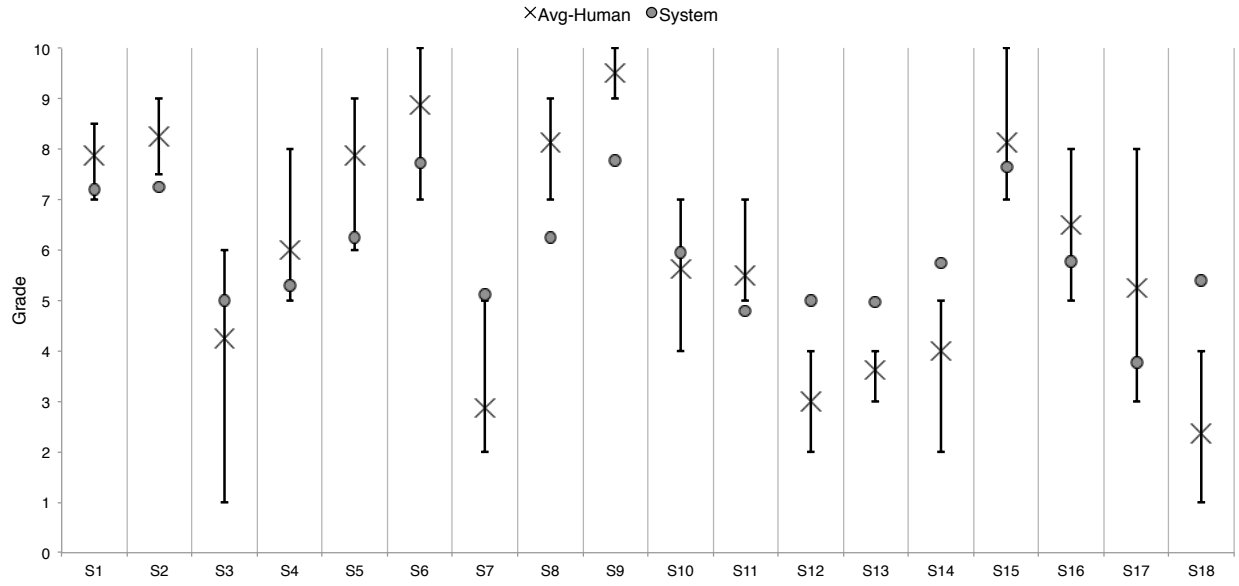


Figure II-21 Correlation of human grades with those of PROCMARK for $\mathcal{BT}$

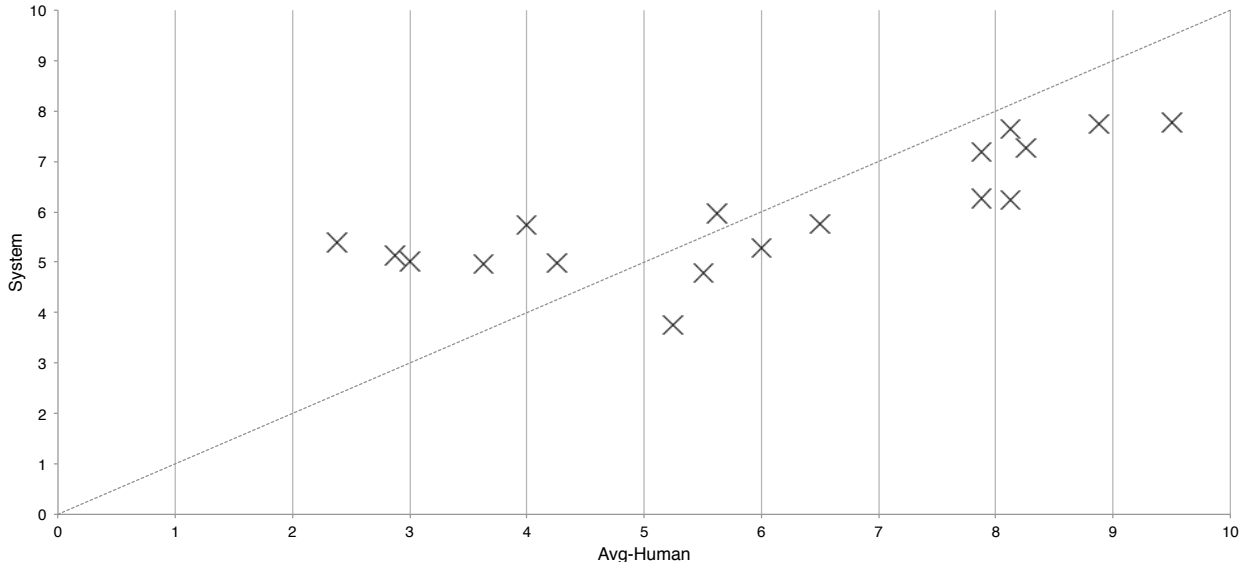
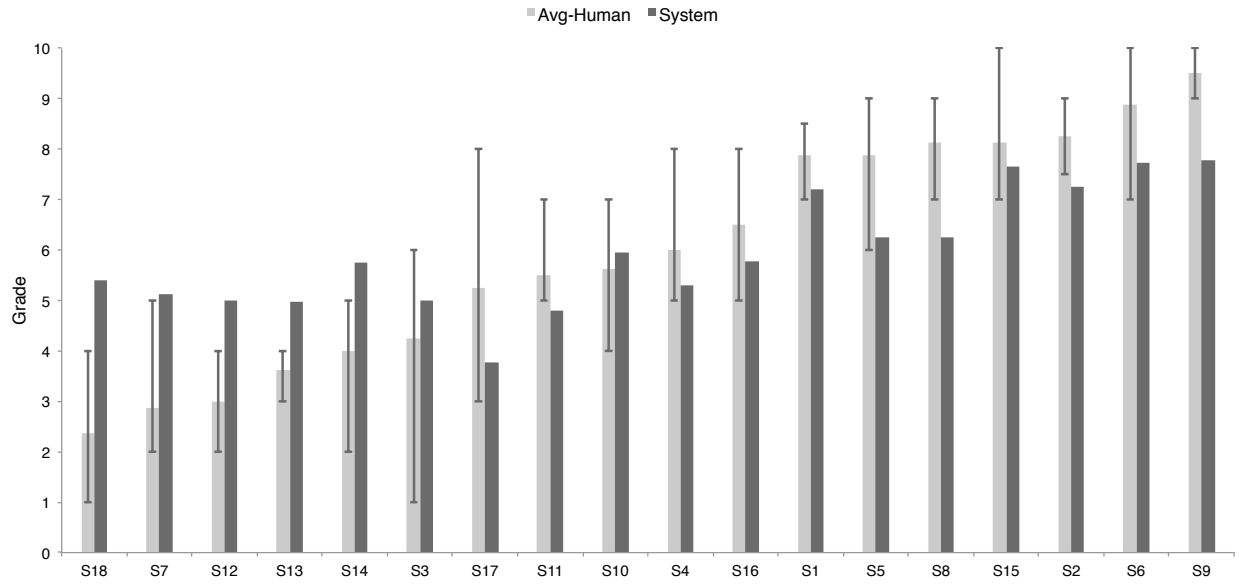


Table II-14 Correlations between pairs of human graders and PROCMARK for $\mathcal{BT}$

	G1	G2	G3	G4	PROCMARK
G1	1.00	**0.64**	0.80	**0.85**	**0.82**
G2		1.00	0.73	0.72	0.64
G3			1.00	0.83	0.60
G4				1.00	0.80
PROCMARK					1.00

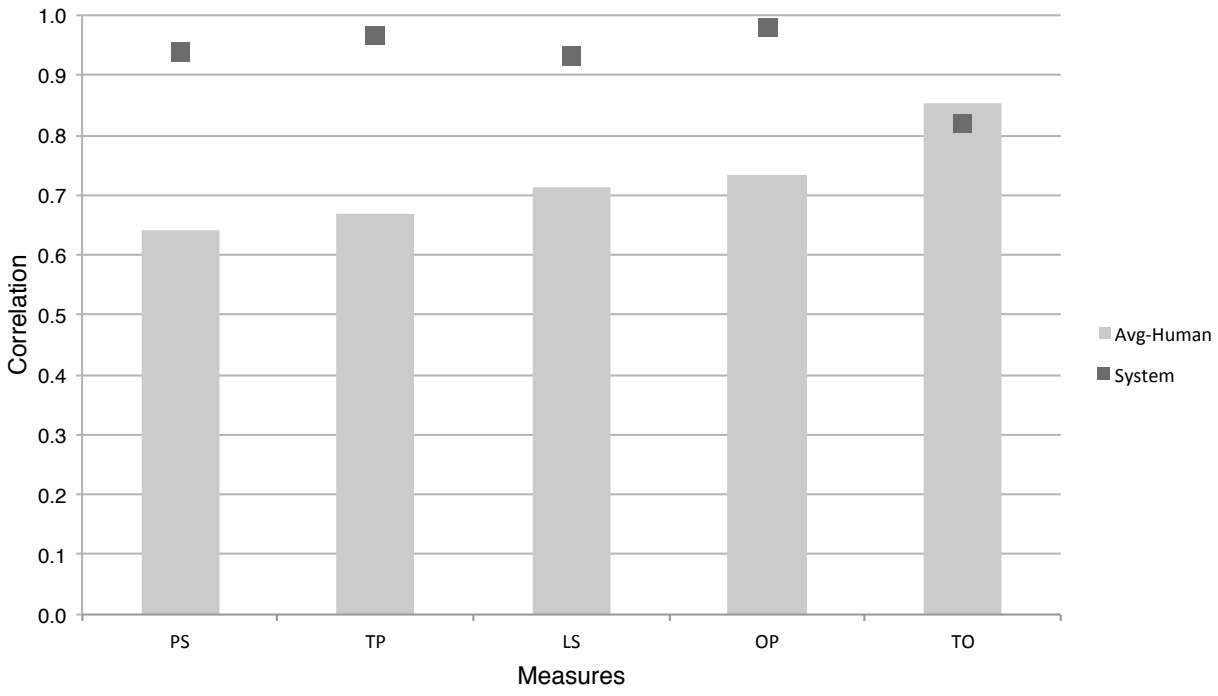
range (out of 10), and slightly less generous grades than humans in the 5 to 10 range (except for S10). Figure II-23 shows the correlation of the individual measures with the average grade given by human graders, and with the final grade assigned by the system. Here, correlation of the TO measure outperforms the correlations obtained for the other algorithms, which leads us to believe this measure might perform better with *less* granular semantic networks.

Figure II-22 Ranking of human grades compared to PROCMARK for $\mathcal{BT}$



Overall, the system gives grades with a positive correlation to human grades. We observe that the system performs well on answers of variable length (except for very short answers) and different granularity levels of the corresponding semantic network. We also observe that the system

Figure II-23 Correlation of individual measures for $\mathcal{BT}$



gives plausible grades, which are not systematically higher or lower than human grades. We also observe that PROCMARK performs more reliably than our first system described in Chapter 1, as the correlations obtained by PROCMARK differ only by 0.09, while those obtained by our earlier version vary from 0.52 to 0.86, which is a significant difference (0.34). We should add that we kept the same weight combinations through all experiments. It would be possible to obtain better correlations by finding the combination of weights that works best for a given algorithm. However, we wanted to achieve a simple, all-purpose system which would perform reliably, with minimal configuration.

5 Conclusion and Future Work

In this paper, we proposed a new system specially designed for the assessment of free-text answers containing procedural knowledge. We evaluated the system using questions from three exams on computer algorithms given at Université de Moncton. A total of 53 students and 7 human

graders participated in our experiments. Overall, the system gives numeric grades with positive correlations (0.70, 0.79, and 0.79) to human grades.

The system introduces several novel ideas. First, to allow the representation of procedural knowledge, we implemented the notion of functional concepts that can be nested and reused. We define a course ontology offering general algorithmic knowledge and choose that all knowledge representation be parameterized as concepts, thus allowing variable granularity levels and the possibility of zooming in and out of these levels. Second, we use semantic networks to represent ideal and student answers. This approach allows the grading system to be language-independent by using an overlay of the semantic network rather than the textual answer. Third, we propose three novel similarity measures adapted to procedural knowledge: part-whole similarity, textual proximity, and temporal ordering.

This approach could be used in other domains where procedural knowledge is central to processing the text. Similar methods are applied in [24, 25] to biomedical ontologies, although not for evaluation. Currently the system is easy to use by Computer Science professors as they are used to graph terminology. We plan on designing a graphical tool to facilitate the edition of semantic networks by non-Computer Science instructors. Also, the system gives only the grade as a feedback to the students. It would be possible to supply a commented semantic network to each student.

Acknowledgments

We thank the Natural Sciences and Engineering Research Council of Canada for partially financing this research, as well as the Canada Foundation for Innovation for financing the infrastructure and the technical support for this work. We also thank the instructors and students who participated in our experiments.

CONCLUSION GÉNÉRALE

Ce travail met de l'avant deux approches, qui amènent la nouveauté de l'évaluation des connaissances procédurales dans les réponses ouvertes des étudiants aux questions d'examen. La première approche enrichit OeLE en ajoutant une nouvelle catégorie de concepts, les concepts fonctionnels, à l'ontologie du cours. Une méta-ontologie permet d'ordonner le sous-ensemble de concepts fonctionnels qui doivent apparaître dans un ordre précis dans les réponses des étudiants.

La seconde propose un nouveau système, PROCMARK, qui représente une approche hybride entre notre première version et MultiNet Working Bench, en apportant plusieurs nouveautés. Nous définissons 10 autres catégories de concepts, ainsi que 15 relations, qui formalisent plusieurs notions générales d'algorithmique dans l'ontologie. Nous utilisons des réseaux sémantiques pour représenter à la fois la réponse idéale (fournie par le professeur) et les réponses des étudiants, qui y sont superposées. Ceci permet non seulement aux étudiants de répondre avec plus ou moins de détails, mais accorde également aux professeurs plus de flexibilité dans la création du réseau sémantique représentant la réponse idéale. De plus, notre approche n'utilise pas de motifs grammaticaux spécifiques à une langue, ce qui permet d'adapter facilement le système à d'autres langues en changeant le dictionnaire. Enfin, nous définissons trois nouvelles mesures de similarité, *part-whole similarity*, *textual proximity* et *temporal ordering*, qui augmentent la robustesse de notre algorithme d'évaluation.

L'évaluation automatique s'effectue à partir du réseau sémantique représentant la réponse idéale fournie par le professeur. Le professeur n'a pas à fournir de réponse idéale textuelle, ni de réponses d'étudiants déjà notées, ni d'exemples de réponses acceptables. De plus, notre système est une amélioration par rapport à [8] du fait que le professeur n'a pas à attribuer de pondération aux (instances de) concepts constituant le réseau sémantique. Notre système infère aussi les annotations manquantes à partir des annotations trouvées, ce qui minimise l'intervention humaine.

Nous comparons l'évaluation automatique de nos systèmes à l'évaluation manuelle (par plusieurs correcteurs) de réponses d'étudiants à trois questions d'examen dans des cours portant sur des

algorithmes informatiques offerts à l'Université de Moncton entre 2011 et 2013. Au total, 53 étudiants et 7 correcteurs humains ont participé à nos expériences. Les étudiants sont appelés à décrire le fonctionnement de deux algorithmes de recherche dans un graphe : la recherche en profondeur, ainsi que la recherche du plus court chemin (Dijkstra). La troisième question consiste à expliquer l'algorithme de recherche d'une valeur dans un arbre B+.

Comme seule consigne, les étudiants sont invités à répondre par des phrases simples, et en langue naturelle (c'est-à-dire en français, et non en code ou en pseudocode). Pour leur part, les correcteurs humains ne reçoivent aucune consigne, sauf d'attribuer une note numérique aux réponses des étudiants selon leurs exigences et critères habituels. Afin de ne pas fausser les résultats, les correcteurs humains n'ont pas eu accès aux notes accordées par le système. Nous comparons les notes données par le système à celles données par les humains en utilisant la moyenne, l'écart-type et la corrélation de Pearson. Nous prévoyons utiliser des statistiques plus avancées pour la comparaison, telles que la corrélation intraclasse [9] dans nos travaux futurs. De plus, nous pourrions tester notre système sur un plus grand corpus.

Notre première approche indique des corrélations positives (0,59, 0,67 et 0,86) entre les notes accordées par le système et celles accordées par les correcteurs humains. Notre seconde approche améliore nos résultats, et les notes octroyées par le système sont positivement corrélées (0,70, 0,79 et 0,79) avec celles des correcteurs humains. Les résultats de notre seconde approche démontrent également que la corrélation des notes du système avec la note moyenne attribuée par les humains surpasse – dans les trois expériences – la corrélation minimale obtenue en comparant les correcteurs humains entre eux (0,39, 0,70 et 0,64).

Il serait intéressant de valider notre système en répétant les expériences sur un plus grand corpus, avec d'autres correcteurs humains, ou en l'appliquant à d'autres algorithmes informatiques. Nous pourrions aussi l'appliquer à d'autres domaines où l'évaluation des connaissances procédurales est primordiale. Citons à titre d'exemple des cours de premiers soins, des cours pour le soutien technique téléphonique, ou encore des techniques d'assemblage qui demandent d'effectuer des tâches dans un ordre précis.

---

9. *Intraclass Correlation Coefficient*, ou ICC.

Il serait également possible d'implémenter une interface graphique qui rendrait la création des réseaux sémantiques plus accessible aux professeurs. De plus, une telle interface permettrait l'évaluation formatrice [4] en guidant les étudiants dans leur apprentissage. Par exemple, une interface graphique pourrait indiquer aux étudiants les parties de leur réponse qui sont incorrectes ou manquantes.

ANNEXE A

ORDRES ATTENDUS POUR $\mathcal{DIJ}$, $\mathcal{DFS}$ ET $\mathcal{BT}$

EXPECTED ORDERINGS FOR $\mathcal{DIJ}$, $\mathcal{DFS}$, AND $\mathcal{BT}$

$\mathcal{DIJ}$

```
WhileNotAllVerticesVisited <is-followed-by> ChooseVertexV
WhileNotAllVerticesVisited <is-followed-by> MarkVAsVisited [inferred]
WhileNotAllVerticesVisited <is-followed-by> ForEachVertexWAdjacentToV [inferred]
WhileNotAllVerticesVisited <is-followed-by> ModifyDvOfW [inferred]
WhileNotAllVerticesVisited <is-followed-by> ModifyParentOfW [inferred]

ChooseVertexV <is-followed-by> MarkVAsVisited
ChooseVertexV <is-followed-by> ForEachVertexWAdjacentToV [inferred]
ChooseVertexV <is-followed-by> ModifyDvOfW [inferred]
ChooseVertexV <is-followed-by> ModifyParentOfW [inferred]

MarkVAsVisited <is-followed-by> ForEachVertexWAdjacentToV
MarkVAsVisited <is-followed-by> ModifyDvOfW [inferred]
MarkVAsVisited <is-followed-by> ModifyParentOfW [inferred]

ForEachVertexWAdjacentToV <is-followed-by> ModifyDvOfW
ForEachVertexWAdjacentToV <is-followed-by> ModifyParentOfW
```

$\mathcal{DFS}$

```
VisitRoot <is-followed-by> VisitFirstChildNode
VisitRoot <is-followed-by> VisitOtherSiblings [inferred]

VisitFirstChildNode <is-followed-by> VisitOtherSiblings
```

$\mathcal{BT}$

```
PerformBinarySearchOnRoot <is-followed-by> FindMinimumKeyGTESearchValue
PerformBinarySearchOnRoot <is-followed-by> IfSearchValueLessThanKey [inferred]
PerformBinarySearchOnRoot <is-followed-by> FollowLeftPointer [inferred]
PerformBinarySearchOnRoot <is-followed-by> FollowRightPointer [inferred]

FindMinimumKeyGTESearchValue <is-followed-by> IfSearchValueLessThanKey
FindMinimumKeyGTESearchValue <is-followed-by> FollowLeftPointer [inferred]
FindMinimumKeyGTESearchValue <is-followed-by> FollowRightPointer [inferred]

IfSearchValueLessThanKey <is-followed-by> FollowLeftPointer
IfSearchValueLessThanKey <is-followed-by> FollowRightPointer

PerformBinarySearchOnNode <is-followed-by> IfLeafContainsSearchValue
PerformBinarySearchOnNode <is-followed-by> ExitWithSuccess [inferred]
PerformBinarySearchOnNode <is-followed-by> ExitWithFailure [inferred]

IfLeafContainsSearchValue <is-followed-by> ExitWithSuccess
IfLeafContainsSearchValue <is-followed-by> ExitWithFailure
```

ANNEXE B

ONTOLOGIE DU COURS UTILISÉE PAR PROCMARK

COURSE ONTOLOGY USED BY PROCMARK

CONCEPTS (114)

```
Action
    ToTraverse
Condition
    DwLessThanDvOfW
    LeafContainsSearchValue
    LeafIsReached
    Maximal
    Minimal
    SearchValueLessThanKey
    VertexIsVisited
        AllVerticesAreVisited
        VIsVisited
        WIsVisited
Element
    Edge
        EdgeFromVToW
    Graph
        Tree
    Key
        MinimumKeyGTESearchValue
    SearchValue
    Vertex
        InternalNode
        Leaf
        Root
        UnvisitedVertexOfMinDist
Field
```

```
    Dv
        DvOfV
        DvOfW
    Parent
        ParentOfV
        ParentOfW
    Visited
        VisitedOfVertex
            VisitedOfV
            VisitedOfW
    Weight
        WeightOfEdgeFromVToW
FlowStructure
    Iterative
        ForEach
        Until
        When
        While
    Selective
        If
    Sequential
        AndThen
        ExitWithFailure
        ExitWithSuccess
        OrThen
FunctionalConcept
    B+TreeSearch
        FollowLeftPointer
        FollowRightPointer
        PerformBinarySearch
    BinarySearch
    DepthFirstSearch
        VisitFirstChild
        VisitOtherSiblings
        VisitRoot
    Dijkstra
```

```
        ChooseVAndThenVisitAdjacent
            CalculateDwAndThenModify
                CalculateDw
                    Dw=DvOfV+WeightOfEdgeFromVToW
                ModifyDvOrThenParent
                    ModifyDvOfW
                        DvOfW=Dw
                    ModifyParentOfW
                        ParentOfW=V
            ChooseVertexV
                V=UnvisitedVertexOfMinDist
            MarkVAsVisited
                VisitedOfV=true
Literal
    False
    Null
    One
    True
    Zero
Operator
    Arithmetic
        Addition
        Assignment
        Division
        Multiplication
        Subtraction
    Comparison
        EqualTo
        GreaterThan
            GreaterThanOrEqual
        LessThan
            LessThanOrEqual
    Logical
        AND
        NOT
        OR
```

```
    Member
        Dereference
Quantifier
    ForAll
    ThereExists
Set
    Edges=E
    Keys
    Vertices=V
        AdjacentVertices
        UnvisitedVertices
Variable
    Dw
    V
    W
```

RELATIONS (17)

```
action
    { } => { }

condition
    { Quantifier, Selective, Until, When, While }
    => { Condition, Logical, Quantifier }

decomposition
    { } => { }

destination
    { Edge } => { Variable, Vertex }

do
    { FlowStructure } => { FlowStructure, FunctionalConcept }
    doAfter
        { FlowStructure } => { FlowStructure, FunctionalConcept }
    doBefore
```

```
        { FlowStructure } => { FlowStructure, FunctionalConcept }
    elseDo
        { } => { }

field
    { Dereference } => { Field }

in
    { Condition, ForEach, Quantifier } => { Set }

instance
    { AdjacentVertices, ForEach, Quantifier } => { Variable }

object
    { Dereference } => { Element, Variable }

operand
    { Arithmetic, Assignment, Comparison, Condition,
      Dereference, Logical, Quantifier }
    => { Arithmetic, Assignment, Comparison, Condition,
         Dereference, Logical, Quantifier, Variable }

source
    { Edge } => { Variable, Vertex }

suchAs
    { Condition, ForEach, Quantifier }
    => { Condition, Quantifier }

to
    { AdjacentVertices } => { Vertex }

value
    { Maximal, Minimal } => { Dereference, Field, Variable }
```

ANNEXE C

QUESTIONS POSÉES AUX ÉTUDIANTS

EXAM QUESTIONS (TRANSLATED)

$\mathcal{DIJ}$

Dans vos propres mots, et par des phrases simples, décrivez le fonctionnement de l'algorithme de Dijkstra. Décrivez comment les données de traitement dv, $parent$ et vu sont modifiées par l'algorithme (vous n'avez pas à expliquer ce qu'elles représentent). Expliquez comment le prochain sommet v est choisi (sans optimisation), et donnez la condition d'arrêt de l'algorithme (en supposant que le graphe est connexe).

> *Translation:* In your own words, using simple sentences, describe Dijkstra's shortest path algorithm. Indicate how the dv, $parent$, and $seen$ fields are modified by the algorithm (without explaining what they represent). Describe how the next vertex v is chosen (without optimization), and give the termination condition of the algorithm (supposing that the graph is connected).

$\mathcal{DFS}$

Décrire la recherche en profondeur dans un graphe.

> *Translation:* Describe Depth-First Search.

$\mathcal{BT}$

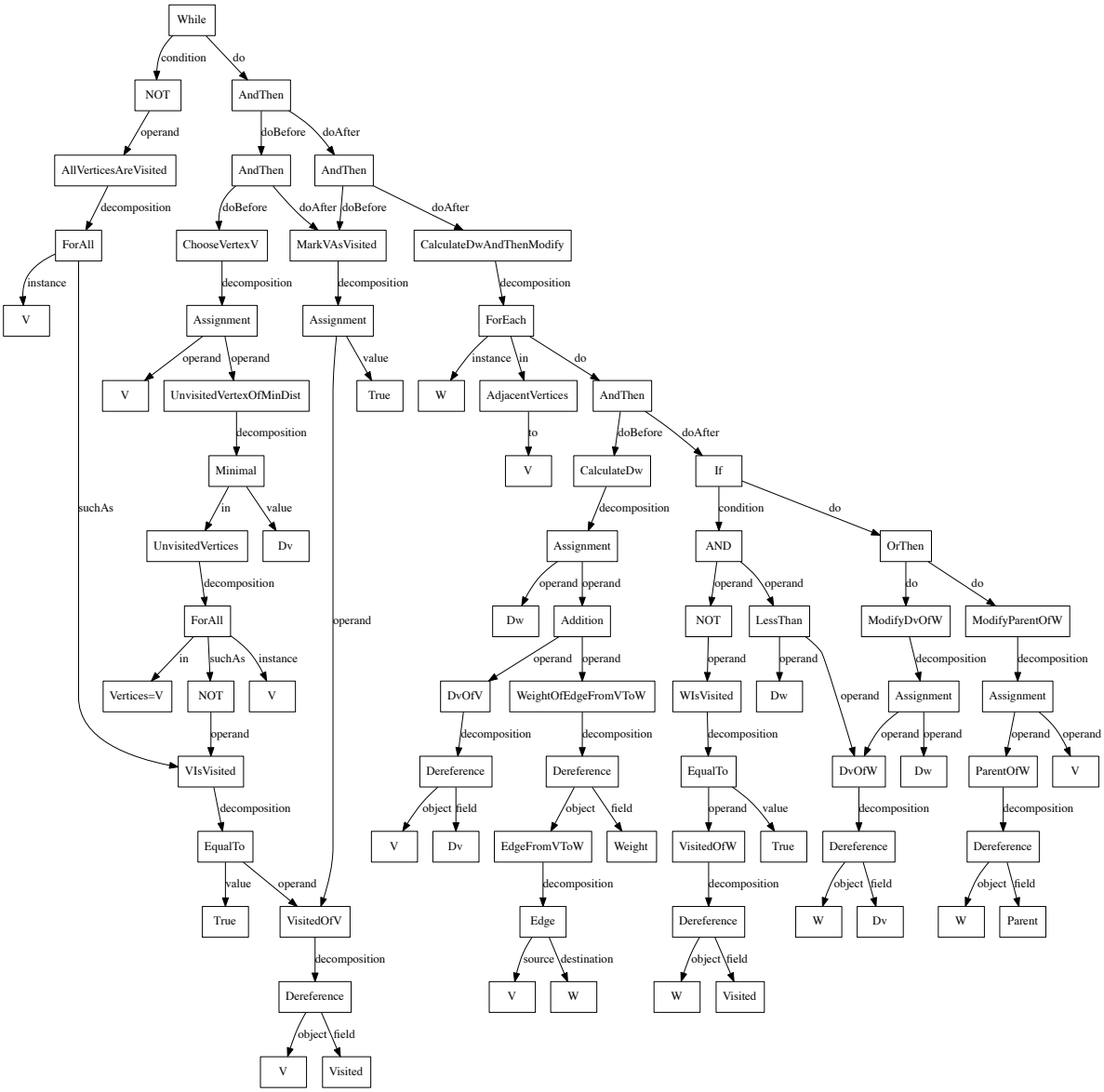
Dans vos propres mots, et par des phrases simples, décrivez le fonctionnement de l'algorithme de recherche d'une valeur dans un B+ Arbre. N'expliquez pas ce qu'est un B+ Arbre, mais seulement comment la recherche d'une valeur est effectuée. Pour votre explication, vous pouvez utiliser des termes comme : noeud, noeud(s) interne(s), feuille(s), racine, valeurs, clés, ...

Translation: In your own words, using simple sentences, describe the algorithm for searching a value in a B+ tree. Do not describe the B+ tree data structure itself, but do indicate how the search is performed. The terms: node, internal node, leaf, root, value, key, ... can be used in your answer.

ANNEXE D

RÉSEAU SÉMANTIQUE DE LA RÉPONSE IDÉALE POUR $\mathcal{DIJ}$

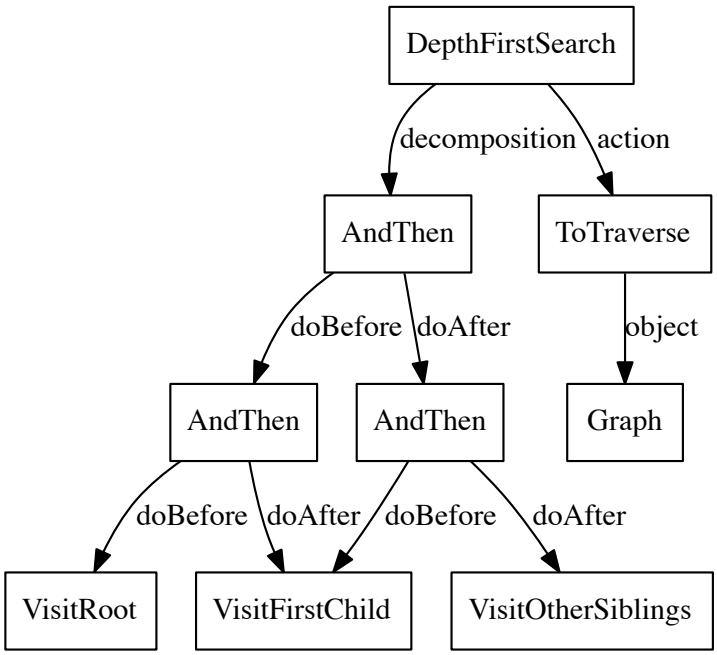
SEMANTIC NETWORK OF IDEAL ANSWER FOR $\mathcal{DIJ}$



ANNEXE E

RÉSEAU SÉMANTIQUE DE LA RÉPONSE IDÉALE POUR $\mathcal{DFS}$

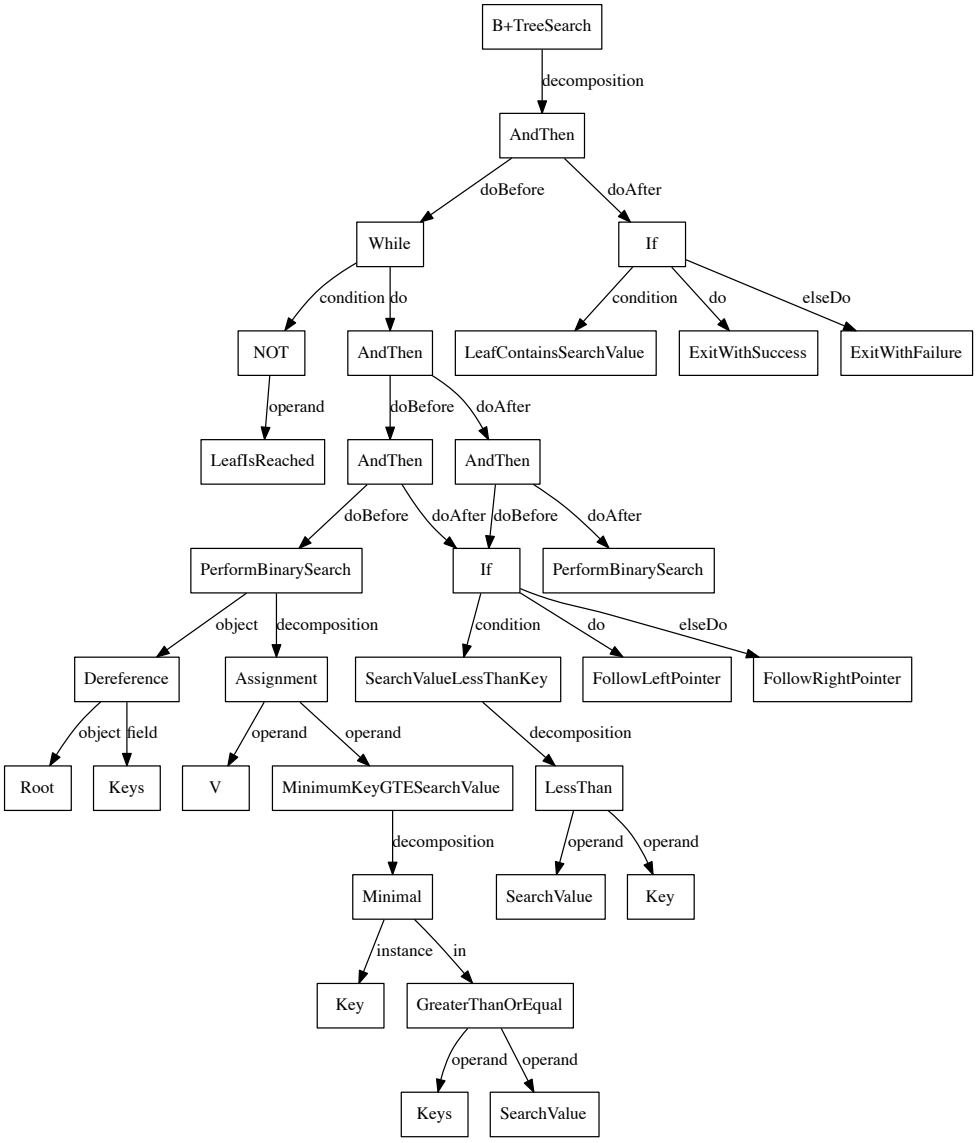
SEMANTIC NETWORK OF IDEAL ANSWER FOR $\mathcal{DFS}$



ANNEXE F

RÉSEAU SÉMANTIQUE DE LA RÉPONSE IDÉALE POUR $\mathcal{BT}$

SEMANTIC NETWORK OF IDEAL ANSWER FOR $\mathcal{BT}$



ANNEXE G

DICTIONNAIRE DE SYNONYMES (D1) UTILISÉ PAR PROCMARK

THESAURUS (D1) USED BY PROCMARK

abandonner (V)
	abandon (N)

aboutir (V)
	aboutissement (N)

accumuler (V)
	accumulation (N)

adapter (V)
	adaptation (N)

afficher (V)
	affichage (N)

agrandir (V)
	agrandissement (N)

ajouter (V)
	ajout (N)
	insertion (N)
	insérer (V)

ajuster (V)
	ajustement (N)

allonger (V)
	allongement (N)

améliorer (V)
	amélioration (N)

ancêtre (N)
	ancestral (A)

appeler (V)
	appel (N)

appliquer (V)
	application (N)

apporter (V)
	apport (N)

approfondir (V)
	approfondissement (N)

arborescence (N)
	arborescent (A)

arriver (V)
	arrivée (N)

arrêter (V)
	arrêt (N)
	interrompre (V)
	interruption (N)
	fin (N)

ascendant (N)

ascendant (A)
ascension (N)

assembler (V)
assemblement (N)
assemblage (N)

augmenter (V)
augmentation (N)
grossir (V)
grossissement (N)
grandir (V)
croître (V)
croissance (N)
croissant (A)
accroître (V)
élargir (V)
prolonger (V)
élargissement (N)
accroissement (N)
prolongement (N)

but (N)
goal (N)

calculer (V)
calcul (N)
recalculer (V)

choisir (V)
choix (N)
possibilité (N)
sélectionner (V)
décision (N)

combinatoire (A)
exhaustif (A)

commencer (V)
commencement (N)

comparer (V)
comparaison (N)

composer (V)
composant (N)
composante (N)

comprimer (V)
compression (N)

compter (V)
compte (N)

considérer (V)
considération (N)

constituer (V)
constituant (N)

continuer (V)
continuation (N)

correspondre (V)
correspondance (N)

coûter (V)
coût (N)
coûtant (A)
coûteux (A)
poids (N)

créer (V)
        création (N)

débuter (V)
        début (N)
        départ (N)

décrire (V)
        description (N)

demander (V)
        demande (N)

dénombrer (V)
        dénombrement (N)

dépiler (V)
        dépilement (N)

déplacer (V)
        déplacement (N)
        bouger (V)

descendant (N)
        descendant (A)

descendre (V)
        descente (N)

détecter (V)
        détection (N)

déterminer (V)
        détermination (N)
        décider (V)

développer (V)
        développement (N)

diminuer (V)
        diminution (N)
        réduire (V)
        rapetisser (V)
        raccourcir (V)
        décroître (V)
        rétrécir (V)
        décroissant (A)
        décroissance (N)
        baisser (V)
        raccourci (N)

échouer (V)
    échec (N)

économiser (V)
    économie (N)
    économique (A)

écrire (V)
    réécrire (V)
    écriture (N)

égaler (V)
    équivaloir (V)
    équivalent (N)
    égalité (N)
    équation (N)
    égal (A)

égaliser (V)
    égalisation (N)

éloigner (V)
    éloignement (N)

enchaîner (V)
        enchaînement (N)

engendrer (V)
        engendrement (N)

entrer (V)
        entrée (N)
        entrant (A)

énumérer (V)
    lister (V)
    énumération (N)

espacer (V)
        espacement (N)

essayer (V)
        essai (N)
        tenter (V)
        tentative (N)
        tentatif (A)

estimer (V)
        approximer (V)
        estimation (N)
        estimé (N)

évaluer (V)
    évaluation (N)

éviter (V)
    évitement (N)

examiner (V)
        examen (N)

exister (V)
        existant (A)
        existence (N)

explorer (V)
        exploration (N)

exprimer (V)
        expression (N)

factoriser (V)
        factorisation (N)

faible (A)
        petit (A)

façon (N)
        manière (N)

file (N)
        queue (N)

fils (N)
        enfant (N)
        successeur (A)
        successeur (N)

finaliser (V)
        finalisation (N)

fonctionner (V)
        fonctionnement (N)

formaliser (V)
	formalisme (N)

franchir (V)
	franchissement (N)

frère (N)
	soeur (A)
	soeur (N)
	frère (A)

garantir (V)
	promettre (V)
	promesse (N)
	garantie (N)

garder (V)
	garde (N)

généraliser (V)
	généralisation (N)

générer (V)
	génération (N)
	formation (N)

heuristique (N)
	heuristique (A)

implémenter (V)
	implanter (V)
	implantation (N)
	implémentation (N)

importer (V)
	importation (N)

inclure (V)
	inclusion (N)

initial (A)
	premier (A)

initialiser (V)
	initialisation (N)

instancier (V)
	instanciation (N)

intermédiaire (N)
	intermédiaire (A)

inverser (V)
	inversion (N)
	renverser (V)

isoler (V)
	isolation (N)

jouer (V)
	jeu (N)

lire (V)
	relire (V)
	lecture (N)

liste (N)
	list (N)

logarithme (N)
	logarithmique (A)

maintenir (V)
	maintenance (N)

marquer (V)
	indiquer (V)
	marque (N)

maximiser (V)
	maximum (N)
	maximal (A)
	maximisation (N)
	meilleur (A)

minimiser (V)
	minimisation (N)
	minimum (N)
	minimal (A)

modifier (V)
	changer (V)
	modification (N)

mémoriser (V)
	sauvegarder (V)
	mémorisation (N)

méthode (N)
	méthodique (A)

nécessiter (V)
	requérir (V)
	nécessité (N)

optimiser (V)
	optimisation (N)
	optimal (A)
	optimum (N)

opérer (V)
	opération (N)

ordonner (V)
	ordonnancement (N)

parcourir (V)
	parcours (N)

père (N)
	mère (N)
	mère (A)
	père (A)

pile (N)
	stack (N)

placer (V)
	placement (N)

pointeur (N)
	pointer (V)

pré-requis (N)
	pré-requis (A)

précéder (V)
	précédent (A)
	prédécesseur (A)
	prédécesseur (N)

prédire (V)
	prédiction (N)

préférer (V)
	préférence (N)

prioriser (V)
	priorité (N)

procédure (N)
	procédural (A)

programmer (V)
	programme (N)
	programmation (N)

quantifier (V)
	quantificateur (N)
	quantification (N)
	quantité (N)
	montant (N)

rassembler (V)
	rassemblement (N)

rechercher (V)
	chercher (V)
	recherche (N)
	trouver (V)

redondance (N)
	redondant (A)

remonter (V)
	monter (V)
	montant (A)

répondre (V)
	réponse (N)

répéter (V)
	itérer (V)

somme (N)
	sommation (N)

sommet (N)
	noeud (N)
	état (N)

succès (N)
	réussite (N)

succéder (V)
	suivre (V)

suivant (A)
	prochain (A)

supprimer (V)
	éliminer (V)
	enlever (V)
	suppression (N)
	enlèvement (N)
	élimination (N)

séparer (V)
	séparation (N)

tableau (N)
	array (N)

terminal (A)
	dernier (A)

```
        final (A)

terminer (V)
        finir (V)
        terminaison (N)

texte (N)
        texte (A)
        textuel (A)

travailler (V)
        travail (N)

trier (V)
        classer (V)
        tri (N)

utiliser (V)
        employer (V)
        utilisation (N)
        user (V)
        usage (N)
        emploi (N)

valider (V)
        validation (N)
        valide (A)

vecteur (N)
        vector (N)

visiter (V)
        visite (N)
        voir (V)
```

ANNEXE H

DICTIONNAIRE ONTOLOGIQUE (D2) UTILISÉ POUR $\mathcal{DIJ}$

ONTOLOGICAL DICTIONARY (D2) USED FOR $\mathcal{DIJ}$

```
#Dv
    distance
    distance actuelle
    dv
    chemin

#Parent
    parent

#ParentOfW
    parent de w

#Weight
    poids

#WeightOfEdgeFromVToW
    poids du côté adjacent
    arc -> poids
    poids de l'arc
    poids de l'arc ( v w )

#Visited
    vu

#Minimal
    minimal
    le plus petit
    le plus court

#VertexIsVisited
```

```
    sommet vu
    fils vu

#WIsVisited
    w <négation> vu
    w <négation> être vu
    w qui <négation> être vu

#Vertices=V
    les sommets
    sommets

#AdjacentVertices
    sommets w adjacents
    sommets adjacents
    destinations
    fils
    voisins de

#NOT
    <négation>

#AND
    et que
    et

#Addition
    somme de
    somme du
    plus
    +
    ajouter

#Dereference
    ->
    de
```

```
#Assignment
    choisir
    marquer
    diminuer
    =
    mettre à jour
    changer
    mettre la valeur
    devenir
    affecter
    désigner
    modifier à la baisse
    améliorer
    récupérer
    définir
    calculer
    le placer comme

#EqualTo
    ==
    =
    est égal à
    est

#LessThan
    inférieur à
    inférieur au
    plus petit que
    plus petit
    moins que
    <

#ForAll
    pour tout
    tous les
```

```
#ForEach
    pour tous
    pour chaque

#While
    tant que

#If
    si

#AndThen
    et
    ensuite

#OrThen
    et

#Dw
    distance tentative <variable>
    distance tentative
    distance <variable>
    <variable>

#V
    sommet <variable>
    sommet actuel
    sommet courant
    prochain sommet
    sommet trouvé
    <variable>

#W
    sommet <variable> adjacent
    sommet <variable>
    <variable>
```

```
    noeud au bout de l'arc

#EdgeFromVToW
    arc ( v w )

#Vertex
    sommet

#True
    true
    vrai

#False
    false
    faux
```

ANNEXE I

DICTIONNAIRE ONTOLOGIQUE (D2) UTILISÉ POUR $\mathcal{DFS}$

ONTOLOGICAL DICTIONARY (D2) USED FOR $\mathcal{DFS}$

```
#Graph
    graphe

#Tree
    arbre

#DepthFirstSearch
    recherche en profondeur
    cette recherche
    recherche

#ToTraverse
    parcourt
    examine

#VisitRoot
    on part de la source
    commence par la racine
    en partant de la racine
    on commence par les noeuds avec profondeur zéro

#VisitFirstChild
    visiter le premier fils d'un noeud
    développement du premier fils d'un noeud
    les fils d'un noeud sont visités
    on va à son premier fils
    visitant le fils de gauche
    on engendre les fils des noeuds
    parcourir les sommets adjacents à un noeud
    visite les enfants
```

```
    parcourt tous les fils
    recherche des fils

#VisitOtherSiblings
    visiter ses frères
    son frère
    on remonte pour aller au frère du fils
    visiter les frères
    revient au frère
    visiter le fils de droite
    parcourir les sommets du même niveau
    les frères

#AndThen
    avant de
    avant
```

ANNEXE J

DICTIONNAIRE ONTOLOGIQUE (D2) UTILISÉ POUR $\mathcal{BT}$

ONTOLOGICAL DICTIONARY (D2) USED FOR $\mathcal{BT}$

```
#While
    jusqu'à ce que
    jusqu'au
    jusqu'à
    parcourir tout l'arbre en profondeur
    parcours de l'arbre
    parcourir tout le tableau ou l'arbre

#NOT
    <négation>

#LeafIsReached
    atteinte d'une feuille
    il arrive aux feuilles
    niveau des feuilles
    on arrive à une feuille
    on se retrouve à la feuille

#BinarySearch
    recherche binaire
    algorithme de recherche rapide

#PerformBinarySearch
    recherche binaire est effectuée
    recherche sur la liste triée de clés

#SearchValue
valeur recherchée
    valeur attendue
    valeur
```

```
    k

#LessThan
    <
    inférieur
    plus petite

#GreaterThanOrEqual
    > =
    valeur attendue ou une valeur plus grande
    supérieur ou égal
    supérieur
    égal ou entre la première et deuxième valeur
    plus grand

#Key
    clé
    élément

#Minimal
    la plus petite

#Root
    la racine
    le premier noeud
    racine
    noeud racine

#Keys
    les clés
    les éléments

#If
    si

#FollowLeftPointer
```

```
    le noeud de gauche est exploré
    passe vers le fils i
    descend vers le fils gauche
    emprunte le sous-arbre de gauche
    recherche le noeud à gauche
    son fils gauche
    ou gauche
    sont à gauche

#FollowRightPointer
    celui de droite
    vers le fils i + 1
    descend vers le fils droit
    emprunte le sous-arbre de droite
    on cherche le deuxième noeud de niveau inférieur
    suit le pointeur vers son fils droit
    va être droite
    se trouvent à droite

#LeafContainsSearchValue
    trouve la valeur dans la feuille
    on trouve k
    l'élément est trouvé
    on l'a trouvée

#ExitWithSuccess
    cela est un succès
    il la retourne
    la recherche est fructueuse
    on peut soit retrouver la clé recherchée
    on arrête
    succès

#ExitWithFailure
    échec
    recherche infructueuse
```

```
ou non
élément <négation> existe
```

RÉFÉRENCES

[1] E. Snow, C. Moghrabi, and P. Fournier-Viger, "Assessing Procedural Knowledge in Open-ended Questions through Semantic Web Ontologies," in AOW2012: Proceedings of the 8th Australasian Ontology Workshop, in conjunction with the 25th Australasian Joint Conference on Artificial Intelligence (AI2012), CEUR, Sydney (Australia), December 2012, pp. 86–97.

[2] E. Snow, C. Moghrabi, and P. Fournier-Viger, "Automatic Grading of Open-ended Questions Using Semantic Web Ontologies and Functional Concepts," in Ontologies: Theory and Applications in Information Systems and the Semantic Web, K. Taylor, A. Gerber, T. Meyer, M. Orgun, Eds., in press, 2015, 12 pages.

[3] E. Snow, C. Moghrabi, and P. Fournier-Viger, "Assessing Procedural Knowledge in Open-ended Questions through Semantic Web Ontologies," in ICTAI 2013: Proceedings of the 25th IEEE International Conference on Tools with Artificial Intelligence, IEEE, Washington, D.C., November 2013, pp. 698-706.

[4] G. Durand, La scénarisation de l'évaluation des activités pédagogiques utilisant les Environnements Informatiques d'Apprentissage Humain, Ph.D. dissertation, Université de Savoie, Chambéry (France), 2006.

[5] C. Douce, D. Livingstone, and J. Orwell, "Automatic test-based assessment of programming: A review," Journal on Educational Resources in Computing, vol. 5, no. 3, September 2005.

[6] E. Kutluhan, J. Hendler, and D. S. Nau, "HTN planning: Complexity and expressivity," in AAAI-94, 1994, pp. 1123-1128.

[7] B. S. Bloom (Ed.), M. D. Engelhart, E. J. Furst, W. H. Hill, and D. R. Krathwohl, Taxonomy of Educational Objectives: The Classification of Educational Goals, Handbook 1: Cognitive Domain, New York: David McKay, 1956.

[8] D. Castellanos-Nieves, J. T. Fernández-Breis, R. Valencia-García, R. Martínez-Béjar, and M. Iniesta-Moreno, "Semantic web technologies for supporting learning assessment," Information Sciences, vol. 181, no. 9, May 2011, pp. 1517–1537.

[9] L. W. Anderson (Ed.), D. R. Krathwohl (Ed.), P. W. Airasian, K. A. Cruikshank, R. E. Mayer, et al., A taxonomy for learning, teaching, and assessing: A revision of Bloom's Taxonomy of Educational Objectives (Complete edition), New York: Longman, 2001.

[10] D. Pérez-Marín, I. Pascual-Nieto, and P. Rodríguez, "Computer-assisted assessment of free-text answers," Knowledge Engineering Review, vol. 24, no. 4, December 2009, pp. 353–374.

[11] D. Callear, J. Jerrams-Smith, and V. Soh, "CAA of short non-MCQ answers," in Proceedings of the 5th International CAA Conference, Loughborough University, 2001.

[12] S. Jordan and T. Mitchell, "e-Assessment for learning? The potential of short-answer free-text questions with tailored feedback," British Journal of Educational Technology, vol. 40, no. 2, March 2009, pp. 371–385.

[13] J. Sukkarieh, S. Pulman, and N. Raikes, "Auto-marking: using computational linguistics to score short, free text responses," in Proceedings of the 29th IAEA Conference, Philadelphia, 2003.

[14] L. Rudner and T. Liang, "Automated essay scoring using Bayes' theorem," Journal of Technology, Learning, and Assessment, vol. 1, no. 2, June 2002.

[15] C. P. Rosé, A. Roque, D. Bhembe, and K. Vanlehn, "A hybrid text classification approach for analysis of student essays," in Proceedings of the HLT-NAACL '03 Workshop on Building Educational Applications using NLP, Edmonton, 2003, pp. 68–75.

[16] O. Mason and I. Grove-Stephenson, "Automated free text marking with paperless school," in Proceedings of the 6th International CAA Conference, Loughborough University, 2002.

[17] J. Burstein, C. Leacock, and R. Swartz, "Automated evaluation of essays and short answers," in Proceedings of the 5th International CAA Conference, Loughborough University, 2001.

[18] W.-J. Hou, J.-H. Tsao, S.-Y. Li, and L. Chen, "Automatic Assessment of Students' Free-Text Answers with Support Vector Machines," LNCS 6096, 2010, pp. 235–243.

[19] R. Lutticke, "Graphic and NLP Based Assessment of Knowledge about Semantic Networks," in Proceedings of the 12th International Conference on Artificial Intelligence in Education, IOS Press, July 2005, pp. 75–80.

[20] P. Wiemer-Hastings, D. Allbritton, and E. Arnott, "RMT: A dialog-based research methods tutor with or without a head," in Proceedings of the 7th International Conference on Intelli-

gent Tutoring Systems, Springer-Verlag, 2004.

[21] D. Pérez-Marín, E. Alfonseca, P. Rodríguez, and I. Pascual-Nieto, "Willow: Automatic and adaptive assessment of students free-text answers," in Proceedings of the 22nd International Conference of the Spanish Society for the NLP (SEPLN), Zaragoza (Spain), September 2006, pp. 367–368.

[22] R. Klein, A. Kyrilov, and M. Tokman, "Automated Assessment of Short Free-Text Responses in Computer Science using Latent Semantic Analysis," in ITiCSE '11: Proceedings of the 16th Annual Conference on Innovation and Technology in Computer Science Education, ACM, New York, 2011, pp. 158–162.

[23] L. Aroyo and D. Dicheva, "Courseware authoring tasks ontology," in Proceedings of the International Conference on Computers in Education, 2002, pp. 1319–1320.

[24] B. Smith, W. Ceusters, B. Klagges, J. Köhler, A. Kumar, et al., "Relations in biomedical ontologies," Genome Biology, vol. 6, no. 5, April 2005.

[25] S. Schulz, K. Markó, and B. Suntisrivaraporn, "Formal representation of complex SNOMED CT expressions," BMC Medical Informatics and Decision Making, vol. 8, no. 1, October 2008.

[26] J. Hobbs and F. Pan. (2006, Sept.). Time Ontology in OWL. World Wide Web Consortium (W3C). [Online]. Available: http://www.w3.org/TR/2006/WD-owl-time-20060927/, last accessed 2014-11-30.

[27] J. T. Fernández-Breis, R. Valencia-García, D. Cañavate-Cañavate, et al., "OeLE: Applying ontologies to support the evaluation of open questions-based tests," in Proceedings of the KCAP'05 WORKSHOP, SW-EL'05: Applications of Semantic Web Technologies for E-Learning, Banff (Canada), October 2005.

[28] S. G. Pulman and J. Z. Sukkarieh, "Automatic Short Answer Marking," in Proceedings of the 2nd Workshop on Building Educational Applications Using NLP, ACL, 2005, pp. 9–16.

[29] R. B. Clariana and E. M. Taricani, "The Consequences of Increasing the Number of Terms Used to Score Open-ended Concept Maps," International Journal of Instructional Media, vol. 37, no. 2, 2010, pp. 163–172.

[30] S. Kiritchenko, S. Matwin, R. Nock, and F. Famili, “Learning and Evaluation in the Presence of Class Hierarchies: Application to Text Categorization,” LNCS 4013, 2006, pp. 395–406.

[31] M. Mohler, R. Bunescu, and R. Mihalcea, “Learning to Grade Short Answer Questions using Semantic Similarity Measures and Dependency Graph Alignments,” in Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL): Human Language Technologies, ACL, June 2011, pp. 752–762.